Sensor-based Collision Avoidance System for the Walking Machine ALDURO

Von der Fakultät für Ingenieurwissenschaften, Abteilung Maschinenbau der

Universität Duisburg-Essen

zur Erlangung des akademischen Grades

DOKTOR-INGENIEUR

genehmigte Dissertation

von

Jorge Audrin Morgado de Gois
aus
Rio de Janeiro - Brasilien

Referent: Prof. Dr.-Ing. habil. Dr. h.c. Manfred Hiller
Korreferenten: Prof. Dr.-Ing. Andrés Kecskeméthy
Prof. Dr.-Ing. Max Suell Dutra

Tag der mündlichen Prüfung: 26.10.2005

Table of Contents

List of Symbols

In this list, the symbol # will be used as a dummy variable to indicate indexes and references to other symbols.

$\mu_{\#}$	Membership Function of set #
$\hat{*}$	T-Norm (Triangular Norm)
$\check{*}$	S-norm (Triangular Co-Norm)
c(#)	Complement of set #
$\therefore$	Therefore (first published in 1659 in the original German edition of *Teusche Algebra* by Johann Rahn)
$p(\# \mid \#')$	Conditional probability density function of event # given the event #'
$p_{joint}(\#\|\#',\#'')$	Joint conditional probability density function of event # given events #' and #"
Occ	Occupancy state *Occupied*
Emp	Occupancy state *Empty*
$m(\#)$	Certainty values of proposition #
$\oplus$	Dempster's rule of combination
ξ	Confidence in map estimation by a neural network
γ	Adiabatic constant for an ideal gas
$\mathcal{R}$	Gas constant
$\mathcal{M}$	Gas molecular mass
T_a	Actual temperature
T_c	Calibration temperature
$I_{\#}$	Intensity of pulse at distance #
G	Polar gain function of sensor receiver
r	Range

α	Azimuth angle
α_{max}	Maximal azimuth angle
β	Rotation angle
d	Measured distance
ε	Range measurement error
L	Maximal setup range
$\hat{G}$	Approximated gain function
S	Set of points in sensor workspace
S_{xy}	Projection of S onto XY plane
$g(\#)$	Approximation polynomial for the membership as function of height #
$w_{\#}$	Firing rate of rule #
$W_{\#}$	Normalized firing rate of rule #
$\mathrm{f}_{\#}$	Output function of rule #
C_{ij}	Cell of position (i, j)
Q_{ij}	Node of position (i, j)
N	Number of measurements considered
P	Design Matrix
Θ	Parameters Set
$\theta_{\#}$	Parameter of output function $\mathbf{f}_{\#}$
$K_{\#}$	Reference frame
$\#_G$	Relative to the grid reference system
$\#_Q$	Relative to the quasi-global reference system
$\#_P$	Relative to the platform fixed reference system
$M_{\#}$	Map in reference system #
h_{leg}	Maximal possible distance from the platform to the foot

h_{sec}	Highest possible position for the foot with bent knee
h_R	Nominal distance from body to ground
h_{max}	Maximal height possible to overcome
h_{min}	Minimal height possible to step into
h_T	Vertical workspace of the foot
ρ	Perception value
ρ^V	Perception vertical component
ρ^H	Perception horizontal component
$\mathbf{B}$	Selection matrix of perception
$\boldsymbol{\rho}$	Perception vector
$\dot{\rho}^*$	Normalized perception value change rate
$\dot{\boldsymbol{\rho}}^*$	Normalized perception vector change rate
V_{Lmax}	Maximal leg absolute velocity
D	Maximal possible distance on the grid

1 Introduction

In the last two centuries, with the advent of the steam engine and later of the internal combustion engine, the world has seen an incredible development in wheeled vehicles. Due to the relative simplicity of such, systems and especially their good stability, which makes it easy to control and drive them, wheeled vehicles have become increasingly popular, being employed for all kinds of tasks such as transportation, agriculture, recreation and even space exploration.

Nevertheless, these systems have their disadvantages too. In general, they need special paths (rail or paved roads) and present difficulties with working in very steep or very rugged terrain. Usually special vehicles have to be used first, to prepare the terrain and enable other ones to work there. This is not always possible or desirable, such as in a protected forest or in a steep canyon.

In searching for solutions for these problems, old ideas are being considered, such as using legs instead of wheels. The history of walking machines is surprisingly long. Early designs can be traced back to the 18th Century, and towards the end of the 19th century more ambitious designs began to appear with the first two-legged walking machine, designed in 1893 by George Moore. Further designs arose around this time covering quadruped walking machines, which were little more than trucks with legs.

The breakthrough development in the history of walking robots came in 1966, when McGhee and Frank (1968) at the University of Southern California constructed ‘Phoney Pony’ (Fig. 1.1), the first computer controlled walking machine.

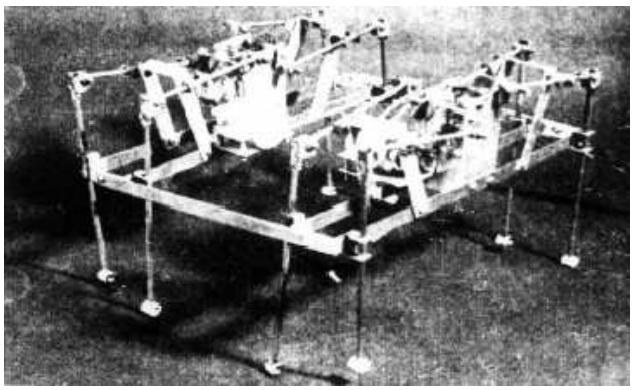

Fig. 1.1: The robot Phoney Pony

The first manual controlled walking truck was the GE Quadruped (Fig.1.2), General Electric Walking Truck by Mosher (1969), finished in 1968. The onboard operator controlled the four legs with levers and pedals associated to his own arms and legs in order to control the robot, what was not easily achieved and made impossible for the operator to execute any task other than leg control.

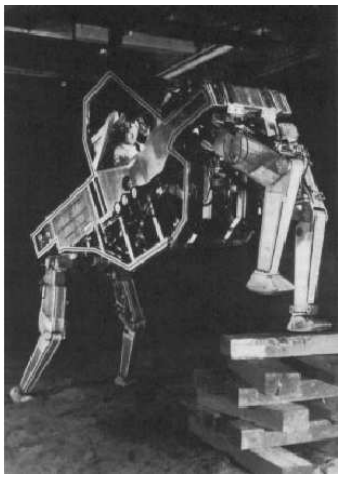

Fig. 1.2: GEE Quadruped

More complex walking machines began to appear in the 1970s, when the level of complexity developed from relatively simple early models to sophisticated ones such as the TITAN models of the 1980s [48]. Whereas all previous legged walkers had concentrated on getting a correct walking action, the TITAN III model took things a step further and incorporated sensors on the feet and a processing system to determine the status of the ground. Developments in robots with more than two legs continue to this day, with emphasis being placed upon advanced navigation methods and increased strength and speed.

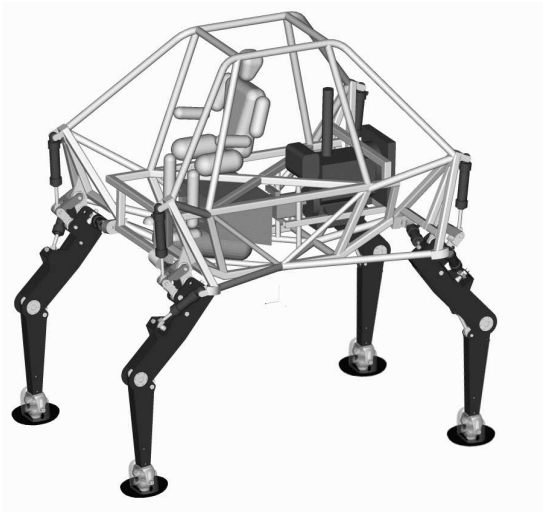

Fig. 1.3: The robot ALDURO

At the Department of Mechatronics at the University Duisburg-Essen the Anthropomorphically Legged and Wheeled Duisburg Robot (ALDURO) is under development [47][90][91]. It is a large-scale four-legged hydraulically driven walking machine with an onboard operator (Fig. 1.3). The machine weighs ≈1600kg, is 3.5m high, is equipped with four identical hydraulically actuated legs and is supposed to operate in very rugged terrain. Operation on smoother terrain enables replacement of the feet of the rear legs with wheels, to increase speed and stability. The general structure resembles that one of the Quadruped, and is similar some mobile excavators designed to operate in steep terrain.

Each of the anthropomorphic legs on ALDURO has four local degrees of freedom, with four independent actuators, which gives a total of sixteen actuators to control. To avoid the problems faced by the operator of the Quadruped, the leg motion generation is automated, enabling control of the robot with a simple joystick. The operator only has to dictate the desired direction of travel and the robot is supposed to execute the leg control autonomously.

1.1 Problem Definition

As ALDURO has an onboard operator, it is not necessary that the robot performs complete path planning. The system actually works like a semi-autonomous navigation system: the driver prescribes the general direction of travel while the robot is responsible for the execution of the gait and other accessory tasks. The control structure has a modular organization, which makes it very easy to add new behaviors in the form of new modules. Such characteristic enables the implementation of complex sensor systems in a relatively simple and elegant way.

As already realized in the TITAN project, to achieve real walking and not only stable gaits on planar surfaces it is necessary for the robot to have information about the ground and surroundings, especially when it controls most of the action. It is then possible for the robot to optimize its actions, achieving stable gaits in even on uneven terrain or to operate in a dynamic environment, where characteristics (such as location of objects) are changing all the time.

ALDURO is intended to operate outdoors, specifically in rough terrain, which constitutes an unstructured environment, i.e. an environment for which there is no map previously available and where most of the objects cannot be classified according to predefined models. Moreover, there will be possibly other machines or people moving in the area of operation, which constitutes a dynamic scenario. Therefore, it is clear that the area where ALDURO will operate is an unstructured dynamic environment.

According to Caurin [1994], since the robot is responsible not only for the execution of the movements, but for most of the planning as well, it is necessary that it is able to recognize the operational area in order to execute its mission without damage to it or other equipment, to installations and specially to people. Such damage is caused by failed path planning, leading either to collision or to a false placement of the feet (e.g. stepping into a hole), which happens because the control system does not have enough information about the environment. Thus, it is clear that it is necessary to provide the robot with a system that supplies up-to-date information about the environment and interprets such information avoiding the above-mentioned dangerous situations.

Such a system is what is generally called a *collision avoidance system*, which is already widely employed in autonomous system navigation and several different techniques already exist. However, given the peculiarity of ALDURO and its employment, existing systems would have to incorporate some features to specifically attend to its characteristics, among which the main ones to be considered are:

- the large dimensions of the robot;
- its relatively low velocity;
- legs and body with spatial movement [55];
- operation in unstructured terrain.

This set of characteristics makes ALDURO a unique application; therefore the use of most existing techniques for collision avoidance would be at least inefficient. Thus the development of a system with a new approach, where such characteristics are the focus is necessary. In this way, when developed, a collision avoidance system oriented toward ALDURO must be able to:

- perform three-dimensional coverage of the environment of the robot, and not only bi-dimensional as is usual;
- generate a relatively small amount of data when considered the space covered, because as a 3D system this amount can easily slow down process, which is not acceptable because such system has to run in real time;
- supply information for, or execute, the necessary movement corrections, taking into consideration that ALDURO is composed of many movable parts, not an only body as most of the mobile robots.

As the central control unit of ALDURO is an onboard PC, the processing capacity is not a so rigid restriction, but if new modules are to be installed on the robot, it is strongly desirable

that each of them demands as small processing time as possible. Another possibility would be the decentralization of the control and sensing activities, which implies the use of microcontrollers, which, in general, have a much lower capacity than a PC.

1.2 Literature Review

Ultrasonic Transducer

Moravec and Elfes (1985) and Elfes (1989) use occupancy grids to represent the spatial information gained from ultrasonic sensors. The occupancy grid is a multidimensional field that maintains stochastic estimates of the occupancy state of the cells in a spatial lattice. To construct a sensor-derived map of the robot's world, the cell state estimates are obtained by interpreting the incoming range readings using probabilistic sensor models. Lim and Cho (1993) and Gourley and Trived (1994) are based on occupancy grids, where a Bayesian model is used to estimate the uncertainty of the sensor information and to update the existing probability map with new range data.

Kuc and Siegel (1987) present a method for discriminating planes, corners and edges using sonar data gathered at two positions. Two significant follow-ups are Barshan and Kuc (1990), which differentiates planes and corners with multiple transducers at a single position, and Bozma and Kuc (1991), which differentiates planes and corners with one transducer at two positions. By using a confidence-based map, Oskard, Hong and Shaffer (1990) incremented or decremented confidence values from an initially assigned base value as confirming or conflicting information is received. This information is integrated with previously available multi-resolution model.

Borenstein and Koren (1989 and 1991) present Histogramic In-Motion Mapping (HIMM) for real-time map building. Like the certainty grid, each cell in the histogram grid holds a certainty value representing the confidence in the existence of an obstacle at that location; however, only one cell in the histogram grid is updated for each range reading. In Schneider, Wolf and Holzhausen (1994) a real time world model based on data from onboard ultrasonic transducers is constructed and statistical methods are used to transform the digital map into a topographical map; a path planner based on the Virtual Force Field method is employed.

Ohya, Nagashima and Yuta (1994) use a system of one ultrasonic transmitter with two receivers to determine the normal of the detected surface. A vector map is used to reconstruct the environment. A similar approach, where form is considered rather than localization is presented in Dudek, Freedman and Rekleitis (1996).

Fuzzy logic is used by Oriolo, Vendittelli and Ulivi (1995) to solve the fundamental processes of perception and navigation. The robot collects data from its sensors, builds local maps and integrates them into the global maps so far reconstructed, using fuzzy logic operators. The inputs to the map-building phase are the range measurements from the ultrasonic sensors. Its Outputs are two updated fuzzy maps. Both convey information about the risk of collision at each point in the environment.

The triangulation technique presented by Wijk, Jensfelt and Christensen (1998), relies on triangulation of sonar readings taken from different positions for estimation of the location of structures in the environment. When a new hypothesis is generated, the corresponding occupancy grid cell is set to a measure of belief that the cell is occupied. In a similar manner, Yi, Khing, Seng, and Wei (2000) used the Dempster-Shafer evidence in sensor fusion with a specially designed sensor model to reduce uncertainties in the ultrasonic sensor. The conflict value in Dempster-Shafer evidence theory is used to modify the sensor model dynamically.

Planar Structured Light

Using structured light, Asada (1990) defines a method for building a three-dimensional world model for a mobile robot from sensory data derived from outdoor scenes. The obstacles are classified according to their geometrical properties such as slope and curvature. The local maps generated by the sensor are integrated into the larger global map. In Little and Wilson (1996), cameras and structured lighting are deployed to capture surface data within the workspace, which is transformed into surface maps, or models. Selected range or distance measurements are used in updating and registering existing CAD models; Delaunay triangulation is used to connect the points.

Laser

Gowdy and Stentz (1990) build a Cartesian Elevation Map (CEM), which is quantified in a two dimensional array, where the content of each element is the height at that point. Various Scanner images are fused by using composition the average elevation values. In Olin and Hughes (1991) color video was used to develop planning software that used digital maps (CEM) to plan a preferred route, and then as the vehicle traversed that route obtain scene descriptions by classifying terrain and obstacles. Later in Klein (1993), a CEM is used with a robot traveling at high speed and avoiding fast movable obstacles. The map is fused by matching the vehicle's pitch, roll and altitude; pre-digitized images are used to eliminate distortion.

Nashashibi, Devy and Fillatreau (1992) used two grid representations: the CEM and a local navigation map, which is a symbolic representation of the terrain (built from the elevation map). The terrain is partitioned into homogeneous regions corresponding to different classes of terrain difficulties, according to the robot locomotion capabilities.

Multiple sensors are used by Kweon and Kanade (1992) for incrementally building an accurate 3D representation of rugged terrain using the locus method, exploiting sensor geometry to efficiently build a terrain representation from multiple sensor data. Such rugged natural shapes are represented, analyzed, and modeled using fractal geometry by Arakawa and Krotkov (1993), where fractal Brownian functions are used to estimate the fractal dimensions, using data from a laser range finder. Later in Krotkov and Hoffman (1994), local models are constructed at arbitrary resolutions, in arbitrary reference frames. These local maps are interpolated without making any assumptions on the terrain shape other than the continuity of the surface.

Three navigation modes are used by Lacroix, Chatila, Fleury, Herrb and Simon (1994) to perform cross-country autonomous navigation: 2D planned navigation mode when the terrain is mostly flat; a 3D planned navigation mode when the area is uneven; and a reflex navigation mode. Hancock, Hebert and Thorpe (1998) present a method for obstacle detection for automated highway environments. It is shown that laser intensity, on its own, is sufficient (and better) for detecting obstacle at long ranges.

In Castellanos and Tardös (2001) a technique for segmenting the sensor data obtained from a mobile robot navigating in a structured indoor environment is presented. The localization of the robot in an a priori map is found by application of a constraint-based matching scheme. A time-of-flight laser ranging system is used by Mäzl and Pfeucil (2000) and is combined with vehicle Odometry to generate a 2D polygonal approximation of an indoor environment. Plaza, Prieto, Davila, Poblacion and Luna (2001) describe an obstacle detector system consisting of the estimation of the distance based on the received power of the reflection of laser beam from an obstacle.

Camera and Stereo Vision

A method for representing a global map consisting of local map representations and relations between them has been developed by Asada, Fukui and Tsuji (1990). First 3D information of the edges of the floor is obtained for each sensor map by assuming the camera model and the flatness of the floor. In Hoover and Olsen (2000), the idea is that mobile robots working in the area tune in to broadcasts from the video camera network (or from environment-based computers processing the video frames) to receive sensor data. The occupancy map is a two-dimensional raster image, uniformly distributed in the floor-plane. Waxman, LeMoigne, Davis, Srinivason, Kusher, Liang and Siddaligaiah (1987) developed a system implemented as a set of concurrent communicating modules and used to drive a camera over a scale model road network on a terrain board.

Using outdoor stereo image sequences, Leung and Huang (1991) developed an integrated system for 3D motion estimation and object recognition. The scene contains a stationary

background with a moving vehicle. Adding another camera, Zhang and Fangeras (1992) use this tri-ocular stereo system to build a local map of the environment. A global map is obtained by integrating a sequence of stereo frames taken when the robot navigates in the environment. A Kalman filter is used to merge matched line segments. Zhu, Xu, Chen and Lin (1994) describe a new framework for detection of dynamic obstacles in an unstructured outdoor road environment by purposely integrating binocular color image sequences. Image sequences are used to determine the motion of a dynamic object.

In Krotkov and Herbert (1995), a binocular head provides images to a normalized correlation matcher that intelligently selects which part of the image to process, and sub samples the images without sub sampling disparities. A real-time approach to obstacle detection is presented by Li and Brady (1998) for an active stereo vision system based on plane transformations. If the transformation matrix for the ground plane can be computed in real-time, it can be used to check if the corresponding points are from the ground plane and hence are not from obstacles. Haddad, Khatib, Lacroix and Chatila (1998) present a probabilistic approach that describes the area perceived by a pair of stereo cameras as a set of polygonal cells. Attributes are computed for each cell to label them in terms of classes, by a supervised Bayesian classifier.

Sensor Fusion

Zimmermann (1993) presents a concurrent behavior control system, where the many different possible behaviors of the system are combined according to the system configuration, in order to emulate an intelligent behavior. In this structure, not the data, but the control efforts are combined in a linear form.

McKerrow (1995) discusses the application of four-level data fusion architecture to ultrasonic mapping with a mobile robot. Perception is defined as a four-step process: detection, recognition, discrimination and response. Jörg (1995) uses heterogeneous information provided by a laser-radar and sonar sensors to achieve reliable and complete indoor world models for both real-time collision avoidance and local path planning. The laser range data is used to incrementally build up a grid-based representation of the environment together with the sonar data. New range information is integrated using an associated weight and frequency measure. The weight expresses the degree of belief that a cell is actually occupied by an obstacle. The frequency measure counts the number of update cycles that have passed since the cell's weight has been incremented last.

Akbarally and Kleeman (1996) present a method of combining sonar and visual data to create a 3D sensing for structured indoor environments. Targets are localized by sonar information and classified appropriately; the visual data is obtained using a grayscale CCD camera and is processed using a Hough transform to extract a set of equations of all lines that occur in the

image. Langer and Jockem (1996) describe an integrated radar and vision sensor system for on-road navigation. Range and angular information of targets from the radar are obtained by Fast Fourier transform. Detected targets are kept in an object list, which is updated by successive data frames from the radar sensor. Target information is fused with road geometry.

Visual data obtained by a binocular active vision system is integrated with ultrasonic range sensors by Silva, Menezes, and Dias (1997) based on a connectionist grid. The cells' values depend on the information about the environment provided by the sensing devices along with its neighbor values. Murphy (1998) uses the weight of conflict metric of the Dempster-Shafer theory to measure the amount of consensus between different sensors. Enlarging the frame of decomposition allows a modular decomposition of evidence.

In Yata, Ohya, and Yuta (2000), information from omni-directional sonar (including distance and angle) and omni-directional vision (providing direction of edges, but not distances) is fused for indoor environment recognition, by extracting environmental features. Leonard and Durrant-Whyte (1991) has used feature-based mapping schemes to identify phantom targets. Data fusion methods associated with feature based mapping include the Kalman filter applied to evidential reasoning (Pagac, Nebot and Durrant-Whyte, 1996).

1.3 Goals and Organization of this Work

Taking into consideration the features listed in section 1.1, a system based on ultrasonic sensors seems to be the most appropriate. Such sensors are quite precise with respect to range measurements, but suffer from intrinsically poor angular resolution, which conversely brings an advantage: they cover the whole volume with each measurement. Because of such inaccuracy, the inverse sensor model plays a relevant role to interpret each measure based on the sensor characteristics. This model tries to extract all information possibly contained in each measure in order to supply to the robot more complete information at a lower cost (less measurements or processing, for example). In order to consider the uncertainties about the measurements and aiming to the simplicity, a fuzzy approach is used to obtain an inverse sensor model.

The information available from the inverse sensor model has to be added in an appropriated way to a base of knowledge, in this case a local map. That is done through a data fusion process, which is the main part of a sensing system, because on it depends the reliability of the knowledge base. Since a fuzzy approach was used for the sensor model, the same is used at the fusion stage and TSK inference system in conjunction to Recursive Least Squares Method is used to perform a fast the data fusion.

As the final stage of the collision avoidance system, a navigation module will be stated, which in a general way, is able to deal with a multi-body system with spatial movement. The navigation module together with the inverse sensor model and the data fusion, which supply it with information about the surrounds, form the group of fundamental components of the collision avoidance system. Their development, implementation and tests are shown and explained through out this work, which is divided in eight chapters and an appendix, as follows:

In Chapter 2 some general concepts about sensing systems are shown, with an approach of an autonomous system viewed as an information structure, composed of many stages, where perception is the one responsible for the sensing activity. The perception is analyzed in its own stages and special attention is paid to the error in measurements. Error, uncertainty and imprecision are discussed, under the aspect of information structure and their importance to the sensor selection. In order to select the appropriate sensor, the main sensor principles are presented, with highlight to sensor systems for distance measurement: ultrasonic sensors, laser range finder and radar. Finally, after analyses of different aspects, the type of sensor to be used is selected.

Chapter 3 reviews the basic principles of Fuzzy Logic that are used throughout this work. An overview on some fundamentals on this subject is brought, where basic concepts that will be necessary in the development of the work are highlighted. The differences between classical and fuzzy sets are explored, as well as between fuzzy set and probability theories. Definitions and terminology are given and operation on fuzzy sets with the use of Triangular Norms and Co-norms is presented. At the end, fuzzy relations are briefly discussed, with emphasis on approximate reasoning and inference rules.

In Chapter 4, a discussion on data fusion is carried out aiming at the selection of an appropriate fusion structure for the collision avoidance system. The different kinds of data fusion are presented, showing the difference and application of each one as single sensor and multiple sensors fusion, as well as their classification in competitive, cooperative and complementary. These are further developed by a presentation and respective analysis of the main approaches to build a data fusion system for robot navigation: Bayesian approach, Dempster-Shafer approach, fuzzy logic and sensor fusion by learning. These approaches are compared to the restrictions imposed by the project, and fuzzy logic is chosen as the most suitable one to implement the data fusion in the collision avoidance system for ALDURO.

Since the fusion approach is defined, it is possible to develop the inverse sensor model, what starts Chapter 5. Because of the chosen approach, special attention is paid to measurement uncertainty of the sensor, what is the basis for the fuzzification of the measurements. Still in this chapter, a common internal representation using elements of the robot kinematics is stated, to provide suitable inputs for the fusion. Fuzzy inference systems are analyzed as a

possibility to perform the fusion, then the main kinds are presented and the TSK system is selected, which will be employed in conjunction with the recursive least squares technique in order to make the update of the inference system parameters. Finally, a hybrid navigation technique is developed to complete the collision avoidance system.

Chapter 6 shows the realization of the system developed in the previous chapter, presenting the selection of the physical ultrasonic sensor and some of its characteristics. This sensor demands the use of I^2C-Bus for communication, which is briefly presented, explaining how it works and the necessary hardware adaptations, like the use of a microcontroller as interface to the main controller. An analysis about the positioning of the sensors on the robot is carried out, stating the number of sensors to be used and the ideal distribution on the robot.

The tests and results of the developed system are shown and interpreted in Chapter 7. First, a simplified version of the collision avoidance module is tested in a 2D environment with help of a small, two degrees-of-freedom wheeled robot. Subsequently, simulations of the three dimensional collision avoidance system are discussed, and finally the results of tests in a test-bank with a real size leg of the robot are shown.

At last, in Chapter 8, the obtained results are commented and conclusions are derived. An outlook of possible future works is outlined.

2 Sensing Systems

Making an autonomous system able to perceive the world in which it operates is a key issue to create adaptive and intelligent behavior. It refers to the process of sensing, the extraction of information from the real world, and the interpretation of that information. Without sensing, an autonomous system could only perform pre-planned actions without any feedback about how well it performs those actions [46]. It could only operate in a static environment because it could not check whether the model of the environment on which actions were planned has changed. Therefore, no unforeseen moving objects or humans could be in the environment and all motion should be performed without errors. To make an intelligent system it should be able to perceive its environment and react to changes in it.

The process of extracting data from environment is called perception. It forms the basis for low-level reactive behavior in autonomous systems and at more sophisticated systems, the information coming from the perception is necessary at higher abstraction levels to build up a map of the surroundings, enabling the robot to locate goals and to find its way through this environment. In this case, the interpreted data is fused with previous data, to acquire or improve a map of the environment and to locate the autonomous system in the map. Many different processes are needed to transform sensed real world data to useful information for an autonomous system.

2.1 Information Structure

An autonomous system depends on several different devices and software modules to execute its mission, receiving and sending data from and to all of them. In this way, the more complicated the whole system is, the more it could be considered as an information process. First, the information about the world is collected, then a decision aiming at the consecution of its mission is taken and finally, actions corresponding to this decision are executed. There exist several definitions for the corresponding information process, but in general this process is usually considered as a flow of information, which could be divided in three stages:

- *Sensing*: starting at a very low level, it comprehends collecting data from environment and to transform it in a suitable form to be employed by the robot reasoning.
- *Reasoning*: based on the information built up at the last step, a decision is taken towards the achievement of the goals of the robot’s mission.

- *Actuation*: as output of the reasoning, abstract commands are generated, which are simplified in order to reach the final actuation device and to act on the momentary environment.

The information process is called flow because the information actually flows: bottom-up at the sensing side of the loop and top-down at actuating side, as shown in Fig. 2.1. The row measurements from sensors are improved, going up on higher levels of abstraction in order to build up information, which is really interesting to the robot. This high abstract information is then used by the reasoning process to decide about the next action, which is given in form of high-abstract commands. Such commands follow the up-down direction: they must be brought to a lower level, to attend different tasks, subsystems and finally directly the device controllers [130].

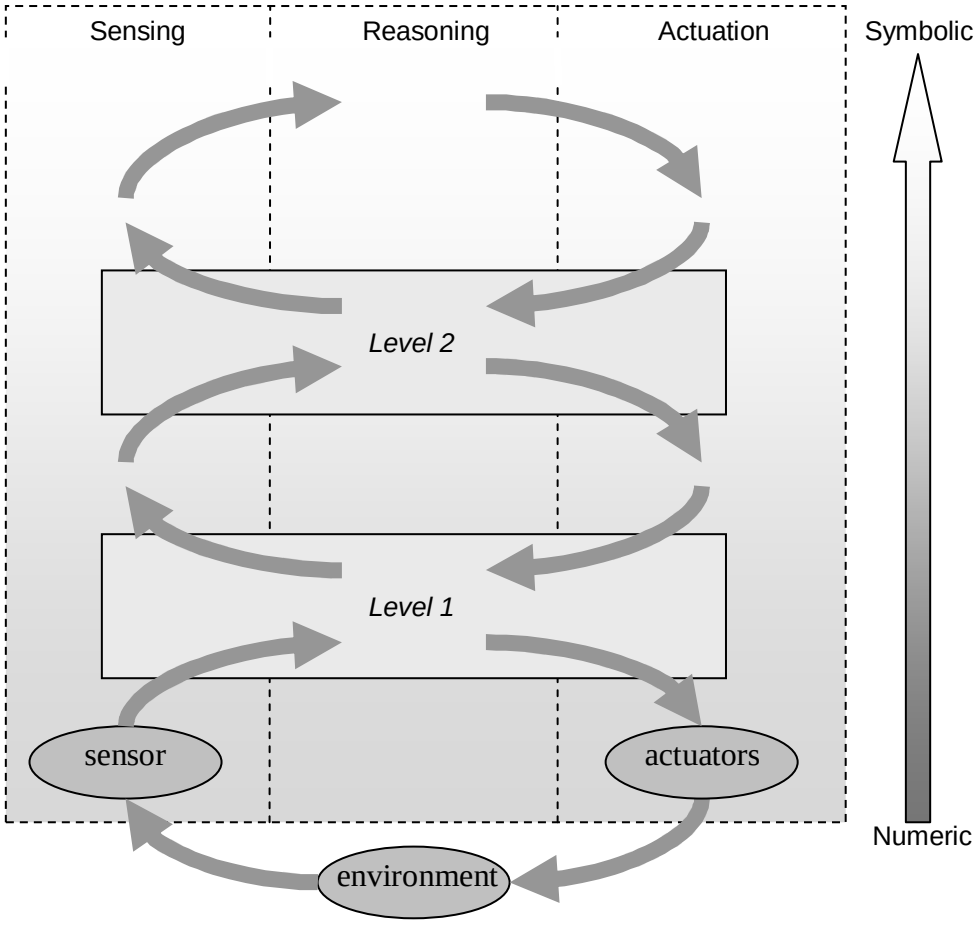

Fig 2.1: Information flow

The mentioned abstraction levels are not always so clear or even do not exist in very simple systems, but in a general way these steps are present in most of the systems.

2.2 Sensing

There is still no standardization about nomenclature and definition of the sensing phases, but here in this work, a structure will be employed that seems to be the most logical and accepted. According to that, the sensing activity can be divided in two distinct modules:

- *Virtual Sensor*: it consists of the whole building process of useful information; starting at the physical measurement (made by the real sensor) of the desired environmental parameter and going up to the conversion of this measured value to a form, appropriated to be fused to a knowledge base.

- *Data Fusion*: with the measured value available in a suitable form, this new piece of information has to be fused to a knowledge base, which contains all the information known by the robot about the environment. Based on this knowledge, decisions will be taken by the reasoning step.

Actually, to fuse a piece of information to knowledge base is not only to add it, because many different aspects of this information have to be considered when fusing it, as the certainty about it. The knowledge base may assume different forms, depending o the kind of data that it contains; in the case of robot navigation that is in general a map of the robot's surroundings. Actually, the base itself is not a part of the sensing, but its result; therefore it is intrinsically related to it, as sketched in Fig. 2.2.

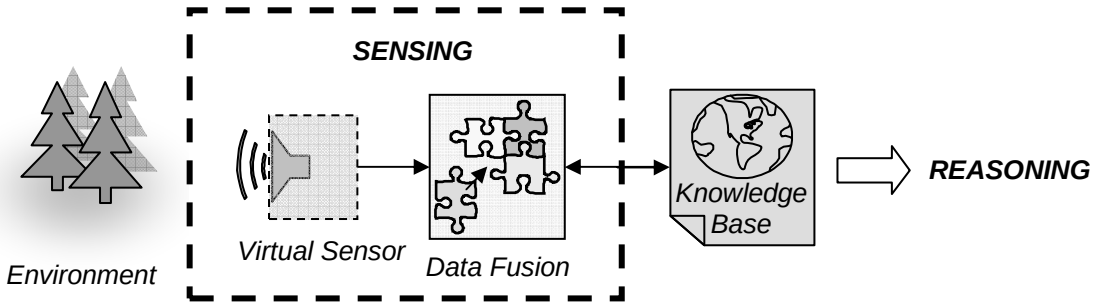

Fig. 2.2: Organization of Sensing

2.2.1 Virtual Sensor and Perception

The virtual sensor is a block responsible for acquiring the necessary data from the environment and providing it in a suitable form; therefore this virtual device is composed of two basic modules, as shown in Fig. 2.3. The first one is the perception, responsible for the acquisition itself and, considering the general case of most autonomous systems, where

interpretation, fusion and reasoning are done by a digital electronic processor, the perception can be organized as:

- *Transducer*: The first component of the whole sensing process is the physical transducer, which transforms a real world quantity into a signal convenient to handle, as for example, an electric signal. The type of sensor needed depends upon the application.

- *AD Conversion*: When working with digital processors, it is necessary to convert the analog signal to a discrete one. This is realized by an analog-to-digital converter or sampler, which quantizes a continuous signal into a digital one that can be read by a computer. The essential parameters of an AD converter are the sampling frequency and the number of bits into which the signal is quantized.

- *Signal Processing*: It can be that the discrete signal is directly usable without any further processing, but usually the output of the discrete signal needs further processing, because it is contaminated with noise, which has to be suppressed, or to filter the desired information out.

The second module of a virtual sensor is the data interpretation, which comprises the exploration of all information contained in the measurement done. In order to be fused, the signal arising from perception has to be interpreted in terms of models of the sensors and / or of the real world, extracting all the necessary information about the parameters of interest. The key issue in such interpretation is the *inverse sensor model*.

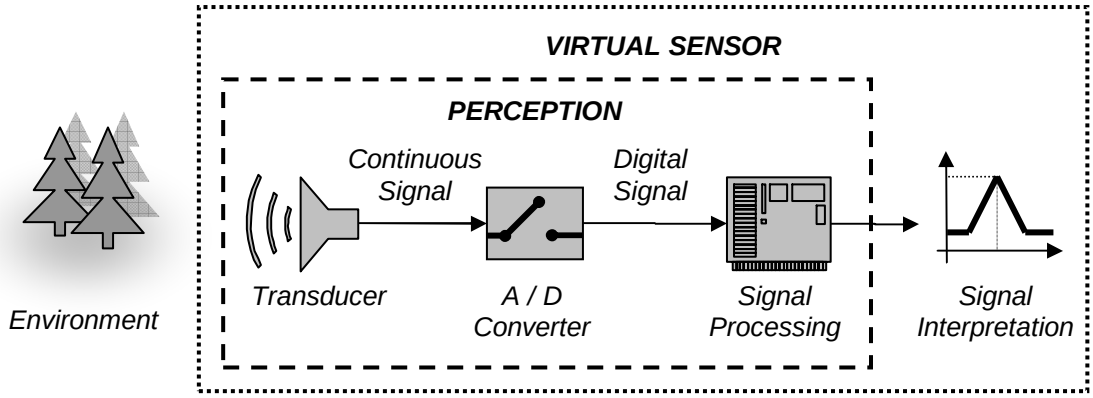

Fig. 2.3: Virtual Sensor components

The inverse sensor model and the data fusion are closely linked, because the former will produce a piece of information from a processed measure, as the later takes this information to improve the knowledge base.

2.2.2 Error, Uncertainty and Imprecision

Signals are mathematically represented as functions of one or more independent variables, however, not always a change in the received signal value corresponds to a change in the real quantity that is supposed to be measured. When the output of a sensor is repeatedly sampled under the same conditions, the output will never be the same, because small variations may be present in the sensor output. These variations result from external disturbances or are due to differences between the true physical operation of the sensing device and the used model.

To interpret the measurements from the transducer, a model of the sensor is needed, but the true physical operation of a transducer is often too complex to model; therefore, the interpretation of sensor measurements often differs from the true physical value of the parameter to be measured. The resulting measurement error (the difference between the measured quantity and the real quantity) is then in general classified as:

- *Systematic Errors*: If a sensor is calibrated under circumstances which differ from those where it is used, a measurement error occurs because an incorrect model or incorrect parameter setting for the sensor was used. The result is that the measurements are systematically wrong. Such kind of error present a certain degree of coherence or even follow a model, they are called systematic errors and may be caused by the wrong choice of parameters or models too, or even by physical failures.

- *Random Errors*: These are characterized by their change when measurements are taken under the same experiment conditions. Random errors can be caused by the way the measurements are taken or by small errors in the sensor itself. Another important source of errors when working with electrical signals are the cables, which are in general very sensitive to disturbances of external electrical fields.

Measurement errors can also be interpreted as ‘uncertainty’ in the sensor measurement: since the sensor is prone to errors, it is uncertain about the true value of the parameter to be measured. This explains why the terms ‘erroneous’ and ‘uncertain’ are often interchanged, but the term ‘uncertainty’ will be used here with the specific meaning of vagueness, while ‘imprecision’ will mean incompleteness in this work. Incompleteness means that a single sensor cannot sense all information, e.g., a single fixed camera cannot view the entire room where a mobile autonomous system is moving in, therefore, multiple views are needed to form a complete view of the room. Measurements can also be incomplete because different sensors are needed to measure all properties of an object.

Uncertainty differs from imprecision in that the latter refers to lack of knowledge about the value of a parameter, usually expressing this value as a crisp tolerance interval, which comprehends the set of all possible values of a parameter. Uncertainty occurs when the interval has no sharp boundaries. Fig. 2.4 shows graphically the difference between the

properties ‘precise’, ‘imprecise’, ‘certain’ and ‘uncertain’ considering a function $\mu(u) \rightarrow [0, 1]$, which describes the possibility of a value to be the real value of the parameter in a universe U.

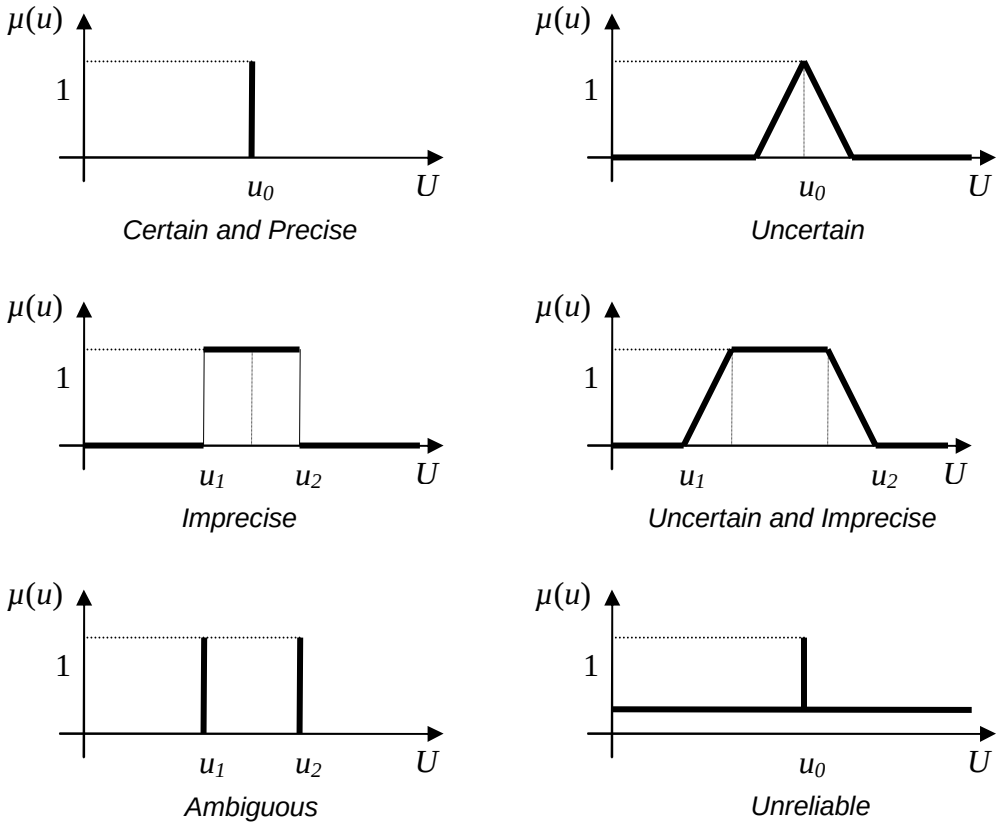

Fig. 2.4: Imprecise and Uncertain values

2.3 Sensor Principles

As long as there is a strict monotone functional relation between the real measured quantity and the generated signal, a suitable model exists that describes this relationship and the sensor can be used. Often the sensing system is more complex and consists of a number of stages in which subsequent processes take place.

A sensor transforms a signal from the outside world into another form, in the case treated here into an electrical signal. Six different domains can be distinguished for these forms of signals from the outside world: radiant signals, mechanical signals, thermal signals, electrical signals, magnetic signals and chemical signals. Each of these different signal domains comprehends a set of physical properties of major importance in relation to the sensing techniques:

- *Radiant Domain*: light intensity, wavelength, polarization, phase, reflectance, transmittance;
- *Mechanical Domain*: position, distance, velocity, acceleration, force, torque, pressure;
- *Thermal Domain*: temperature, specific heat, heat flow;
- *Electrical Domain*: voltage, current, charge, resistance, inductance, capacitance, dielectric constant, electric polarization, frequency, pulse duration;
- *Magnetic Domain*: field intensity, flux density, moment, magnetization, permeability;
- *Chemical Domain*: composition, concentration, reaction rate, toxicity, pH.

Conversion from one signal domain to another is based on one of the many existing physical and chemical effects, which originate many different measurement principles. Signals are carried by some form of energy and sensors transform this incoming energy into electrical energy (the output in case of autonomous systems). If no additional source of energy is needed to obtain the output signal, the sensor is called *self-generating*; otherwise, when an additional energy source is needed for the operation, we call the sensor a *modulating* one, because the energy source is modulated by the measured quantity.

At each signal domain, different principles can be employed to measure the different properties mentioned above. Some of these principles used to create sensors for the five (non-electrical) signal domains could be shortly summarized as follows:

- *Radiant signals*: Electromagnetic radiation includes besides the visible light also radio waves, microwaves, X-rays and gamma rays, which differ in wavelength. Solid-state sensors for (visible) light are mainly based on the photoelectric effect that converts light particles (photons) into electrical charge. Image sensors like CCD cameras are nowadays very cheap and form a rich source of information to access the environment around an autonomous system.
- *Mechanical signals*: There is an important difference between sensors that measure position with and without mechanical contact to the real world. Various physical principles are exploited for measuring position or proximity, including inductive, capacitive, resistive and optical techniques. To measure distances in robotics applications ultrasonic sensors, laser range scanners and radar systems are used. Acceleration, force and pressure are in general not directly measured, but first converted to a displacement. Nowadays, piezoelectric solutions make possible to obtain electrical signal directly from such quantities.

- *Thermal signals*: The resistance of a metal or a semi-conductor depends upon temperature. This relation is well known and is exploited for temperature sensing. In addition, the base emitter voltage of a bipolar transistor is temperature dependent, and is used in many commercially available low-cost temperature sensors. Self-generating temperature sensors can be obtained using the Seebeck-Effect - the so-called thermo-couple.

- *Magnetic signals*: Most of the low-cost magnetic sensors are based on the Hall-effect. When a magnetic field is applied to a conductor, in which a current flows, a voltage difference over the conductor results in a direction perpendicular to the current and the magnetic field. Since this effect is quite substantial in semi-conductors, semi-conductor Hall-plates are low-cost and used in many commercially devices.

- *Chemical signals*: The chemical signal can be directly converted to an electrical signal or first converted into an optical, mechanical or thermal signal, which is then converted into an electrical signal. Many chemical sensors are based on the measurement of the change of the conductivity or the dielectric constant of a chemical when it is exposed to a gas or electrolyte.

2.4 Sensor Systems for Distance Measurement

An objective of this work is to develop collision avoidance system for ALDURO, a sensor system that enables the robot to recognize its surroundings is necessary [5]. In this case, such recognition is done by estimating the position of the surrounding objects through distance measurements. That would be impossible by means of contact sensors because of the huge dimensions of the robot; therefore, no contact sensors using the principle of time of flight were considered. By this principle, the distance is estimated from the traveling time of a pulse, as shown in Fig. 2.5.

The sender (or the transducer switched to send) emits a pulse, which (in the case of an object to lay on its propagation way) it is reflected back. The elapsed time between sending the pulse and the detection of its reflected part by the receiver (or the transducer switched to receive) is measured. This distance is computed by dividing the velocity of pulse in the propagation media by two times the measured time interval. If sender and receiver are not the same device, trigonometric calculations have to be carried out to account for the separation between them, but in general this length is negligible if compared to the measured distance, and the computed follows as sender and receiver were just one.

The combination of the time-of-flight principle and the medias presented in last section made possible the construction of many different sensors for distance measurement without contact.

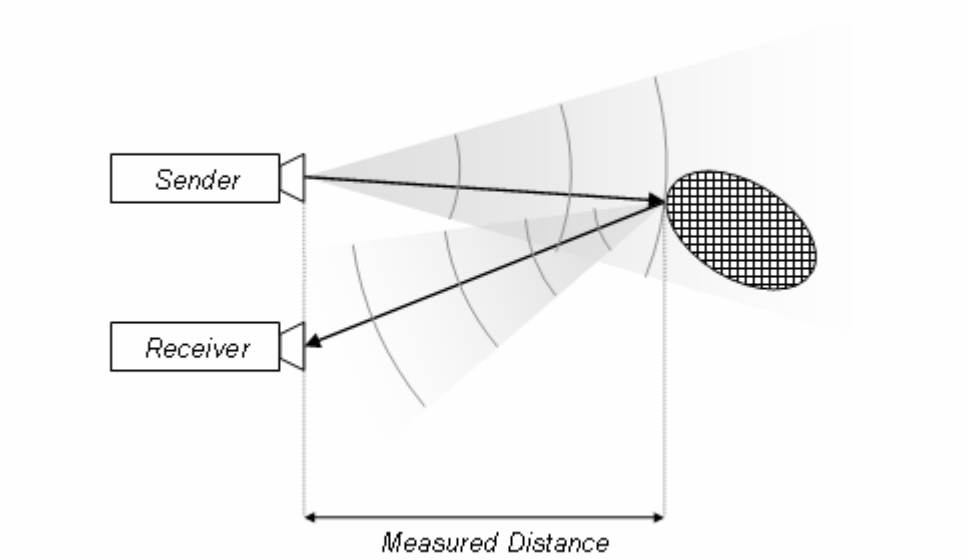

Fig. 2.5: The Time of Flight technique

2.4.1 Ultrasonic Sensors

Very cheap distance measurements can be realized with ultrasonic sensors, resulting in the high popularity of this kind of sensors. Despite their low cost, ultrasonic sensors have some disadvantages as the relatively large opening angle of the cone comprehending their workspace, the cone in which the sound pulse is transmitted and received. Thus, it is possible to realize that there is an object present at a certain distance, but is not possible to estimate the location very precisely. Besides, the sound velocity in air is temperature dependent, therefore changes in temperature influence the measurement introducing a systematic error, that however can be easily avoid through continuous calibration.

Nevertheless, the large cone angle can be an advantage too: a whole volume is covered by single measurement. Moreover, depending on the used frequency, ultrasonic sensors are able to measure the distance to the detected object with good precision. The wavelength of a pulse emitted by a cheap sensor working at 40 kHz is about 8mm, which is good enough for most applications [2]. Combination of sensor readings at different locations or moments (sensor data fusion) is needed to model the environment with ultrasonic sensors.

2.4.2 Laser Range Finder

Laser range finders, also called LIDAR (LIght Detection And Ranging), work by the same principle as ultrasonic sensors, but they emit a light pulse instead of a sound pulse and measure the time-of-flight of the reflected light. As the speed of light is about 300000km/s, time intervals to be measured are in the order of 1-10 nano-seconds, which are measured by an ultra fast clock. Another measurement principle is to modulate the intensity of the laser beam (typically with 5 MHz) and to measure the phase shift of the reflected light. To obtain a 1D range scan of the environment, the laser beam is deflected by a rotating mirror, scanning the environment in a (semi) circle and in the same sense is possible to make a 2D scan.

2.4.3 Radar

A *Radio Detection and Ranging* system emits during a short time a pulse of energy in the radio frequency domain, ranging in wavelength from a few centimeters to about 1m, and uses again the time-of-flight to measure the distance. Continuous-wave radar broadcasts a modulated continuous wave. Signals reflected from objects that are moving relative to the antenna will be of different frequencies because of the Doppler effect. The difference in frequency bears the same ratio to the transmitted frequency as the target velocity bears to the speed of light. In this way, besides the distance also the speed can be measured. Because of the high processing speed achieved by modern solid-state hardware, radar has become an affordable sensor.

2.5 Sensor Selection

As already mentioned, a collision avoidance system requires to know the form and position of the objects around the robot. For the three types presented for distance measurement sensors the corresponding advantages and disadvantages are summarized in Table 2.1.

The precision of distance measurement depends on several factors, as for example, the sample rate (in case of digital output), the internal circuitry, the clock speed and the transducers themselves, but a limiting factor is the pulse wavelength. Since sensors using electromagnetic media can achieve wavelengths of some nanometers, these would have a natural advantage. However, the precision of some millimeters offered by ultra-sound is already enough for this application.

Table 2.1: Comparison of distance measurement sensors

	LIDAR	*Radar*	*Ultra-sound*
Distance Precision	Very high	Very high	High
Direction Precision	Very high	Medium	Low
Overall Velocity	Low	Medium	Medium
Robustness	Low	High	High
Cost	High	Medium	Low
Reliability	Low	Medium	Medium

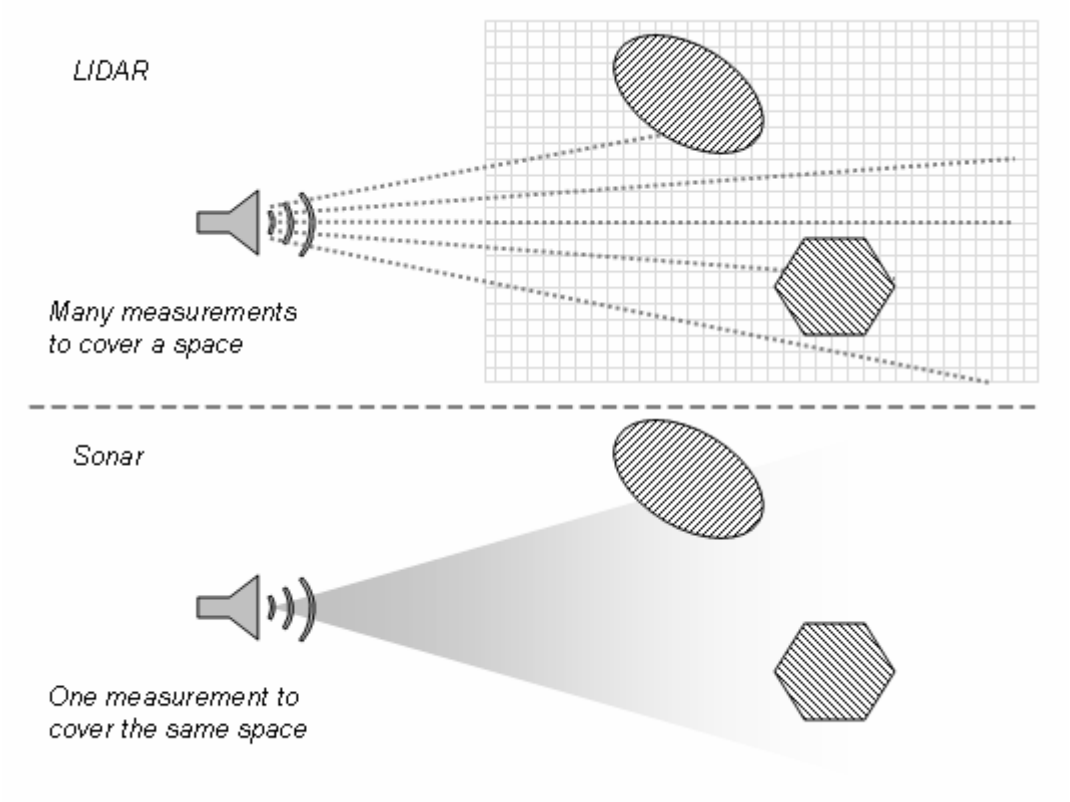

Fig. 2.6: Comparison between spaces covered by LIDAR and Sonar

Regarding the location, the punctual measurements made by LIDAR is an advantage because it gives the precise direction to the point to which the distance is measured; otherwise a broad detection cone (as in radar and ultra-sonic sensors) enables to cover a whole volume with just one measurement.

The LIDAR cannot actually sample a volume, it does that by sampling: the 2D space to be scanned is divided as a mesh and the distance to the node points is measured. The nominal values of the ultrasonic range finder SRF08 are beam angle about 55° and range of 6 m; to scan it with a mesh resolution of 20 cm, a LIDAR would need almost 800 measurements. In spite of using a much faster media, the LIDAR needs much more measurements to cover the same volume as sketched in Fig. 2.6, what makes it slower for an application with so big spaces to cover and much more machine consuming. As a more sophisticated technology, with moveable parts, the LIDAR tends to be less robust than the other ones, which can be very robust depending on the adopted package. Considering cost, the laser range finder is about 10 times more expensive than a radar unit and 100 times that of an ultrasonic rangefinder.

In view of all these characteristics, radar and ultrasound systems seem to be suitable to this application. They have similar characteristics and most of the commercial radar units have detection beams sharper than ultrasonic range finders, but anyway a posterior data fusion is necessary with both of them; then, the decision is taken considering the cost criterion, what leads to the use of ultrasonic range finders.

3 Overview on Fuzzy Logic

Because in the next chapters models will be developed by employing Fuzzy logic, a brief overview on this theory is presented here in order to make the ensuing derivations more understandable.

Fuzzy Logic was primarily designed to represent and reason with some particular form of knowledge. It was assumed that the knowledge would be expressed in a linguistic or verbal form [124], however, when using a language-oriented approach for representing knowledge, one is bound to encounter a number of nontrivial problems. Consider the simple problem of classifying a person as tall. Different ways of representing knowledge about 'tall' persons could be stated; because the property 'tall' is inherently vague, meaning that the set of objects it applies to has in general no sharp boundaries. In this case, it is only to a certain degree that a property is characteristic of an object. This theory was motivated in a large measure by the need for a conceptual framework able to grips with this inherent vagueness.

The fuzziness of a property lies in the lack of well-defined boundaries of the set of objects to which this property applies. More specifically, take a universe of discourse covering a definite range of objects; now consider a subset of this universe, where the transition between membership and non-membership is gradual rather than abrupt. This fuzzy subset obviously has no well-defined boundaries. Then, a membership degree l is assigned to the objects that completely belong to the subset; conversely, the objects that do not belong to it at all are assigned a membership degree of 0. Furthermore, the membership degrees of the borderline cases will naturally lie between 0 and 1. Thus, the use of a numerical scale, as the interval [0,1], allows a convenient representation of the gradation of membership.

3.1 Classical Sets

A classical set is a collection of objects of any kind. The concept of a set has become one of the most fundamental notions of mathematics. The German mathematician Georg Cantor founded the so-called set theory, in which the notions set and element are primitive, which are not defined in terms of other concepts.

A set is fully specified by the elements it contains and the way in which these elements are specified is immaterial. For any element in the discourse universe, it can be unambiguously determined whether it belongs to the set or not. A classical set may be finite, countable or uncountable. It can be described by either listing up the individual elements of the set, or by

stating a property ϖ to define the membership: if $\varpi(u)$ is a predicate stating that u presents the property ϖ, then a set can also be denoted by $\{u \mid \varpi(u)\}$.

Classical set theory uses several operations like complement, intersection, difference, etc. Let A and B be two classical sets in a universe U, then the following set operations can be defined:

- Complement of A, $A' = \{u \mid u \notin A\}$.
- Intersection of A and B, $A \cap B = \{ u \mid u \in A \text{ and } u \in B\}$.
- Union of A and B, $A \cup B = \{ u \mid u \in A \text{ or } u \in B\}$.
- Difference of A and B, $A - B = \{ u \mid u \in A \text{ and } u \notin B\}$.
- Symmetric difference of A and B, $A + B = A - B \cup B - A$.
- Power set of A, $\wp(A) = \{ A^* \mid A^* \subseteq A\}$.
- Cartesian product of A and B, $A \times B = \{(u, v) \mid u \in A \text{ and } v \in B\}$.
- Power n of A, $A^n = A \times A \times \ldots \times A$, n times.

The most important properties of the classical set-theoretic operations are well known as being: Associativity, Distributivity, Idempotence, Absorption, Absorption of complement, Absorption by U (discussion universe) and $\varnothing$ (empty set), Identity, Law of contradiction, Law of excluded middle and the De Morgan's law.

Two ways to define a set were mentioned: by enumeration of its elements or by the definition of a predicate, meaning that every element of the set has a certain property corresponding to the defined predicate. A third way, which is interesting with respect to the extension from classical set theory to fuzzy set theory, is the definition of a set using its characteristic function μ_A as in Eq. 3.1:

$$\mu_A(u) = \begin{cases} 1 \Leftrightarrow u \in A \\ 0 \Leftrightarrow u \notin A \end{cases} \tag{3.1}$$

3.2 Fuzzy Sets

In fuzzy set theory, classical sets are called crisp sets, in order to distinguish them from fuzzy sets, and the membership property is generalized. Therefore, in fuzzy set theory it is not

necessary that an element either belongs to a certain set or not. To any crisp set *A*, it is possible to define a characteristic function μ_A: $U \rightarrow \{0, 1\}$. In fuzzy set theory, the characteristic function is generalized to a membership function that assigns to every element of the discussion universe *U* a value from the unit interval [0, 1] instead from the two-element set {0, 1}. A set *A* whose definition is based on such an extended membership functions is called a fuzzy set. It holds:

$$\begin{aligned} A &= \left\{ (u, \mu_A(u)) \,\middle|\, u \in U \right\} \\ \mu_A &: U \rightarrow [0,1] \end{aligned} \tag{3.2}$$

The definition of a fuzzy set from Eq. 3.2 shows that for such sets, the membership function can assume values between 0 and 1, what means that for an element there is not anymore just the possibilities of belonging or not belonging to a set, but there is an intermediate membership state. That can sound strange when thinking of sets just like groups of elements, but when using the definition of set as 'group of elements presenting a certain property', intermediate membership degree could be viewed as the intensity in which this property is present in each element. Still another definition could be done by using a predicative to define the set, and then the set would be a 'group of elements attending the given predicative'. In this sense, the continuous membership degree describes how much this predicate is fulfilled by each element.

3.2.1 Fuzzy Set Theory versus Probability Theory

In classical probability theory, in a *frequentist* view, an event is defined as a crisp subset of a certain sample space. Furthermore, an event occurs if at least one of the elements of the subset that defines the event occurs. The probability of the event is the proportion of its occurrences in a long series of experiments.

Another type of probability is the *subjective* probability, which is described as the amount of subjective belief that a certain event may occur. Here, a numerical probability only reflects an observer's incomplete (uncertain) knowledge about the true state of affairs. In knowledge representation and inference under uncertainty, it is this concept of probability that is relevant. However, a membership function cannot be viewed as a probability function in order to represent vagueness or fuzziness by means of probability theory [127]. As the elements of the sample space are crisp numbers, when considering the predicate applied to one of them the probability theory requires a response in the set {0, 1} as the fuzzy theory has a response in the interval [0, 1].

A fuzzy set induces a possibility distribution [125] on the universe of discourse, describes how much a predicate is satisfied, while the probability distribution just shows how oft the

event defined by this predicate occurs. Thus, a high degree of possibility does not imply a high degree of probability; however, if an event is probable, it must also be possible.

3.2.2 Basic Definitions and Terminology

Some definitions are important when dealing with fuzzy sets; here the ones of major interest for this work are highlighted.

- *Support* of a fuzzy set *A* is the crisp set that contains all elements of *A* with non-zero membership degree, as presented in Eq. 3.3.

$$\text{support}(A) = \{\, u \in U \mid \mu_A(u) > 0 \,\} \tag{3.3}$$

- *Convexity* means to a fuzzy set that the membership function does not contain 'dips', i.e., it is increasing, decreasing or bell-shaped, as shown in Eq.3.4.

$$\forall u, v \in U,\ \forall \lambda \in [0,1]:\ \mu_A(\lambda \cdot u + (1-\lambda) \cdot v) \geq \min(\mu_A(u),\ \mu_A(v)) \tag{3.4}$$

- *Width* of a convex fuzzy set *A* is defined Eq. 3.5, where the *supremum* operation is denoted as *sup* and the *infimum* operation as *inf*. It represents the size of the interval on the discussion universe, over which the membership function is non-null.

$$\text{width}(A) = \sup(\text{support}(A)) - \inf(\text{support}(A)) \tag{3.5}$$

- *Nucleus* of a fuzzy set A is the crisp set that contains all values with membership degree 1, as defined in Eq. 3.6. If there is only one point with membership degree equal to l, then this point is called the peak value of *A*.

$$\text{nucleus}(A) = \{u \in U \mid \mu_A(u) = 1\} \tag{3.6}$$

- *Height* of a fuzzy set *A* is the largest membership degree μ_A, as defined by Eq. 3.7. A fuzzy set is called normal if hgt(*A*) = 1 and subnormal if hgt(*A*) < 1.

$$\text{hgt}(A) = \sup_{u \in U} \mu_A(u) \tag{3.7}$$

3.2.3 Operation on Fuzzy Sets

Notions like equality and inclusion of two fuzzy sets are immediately derived from classical set theory [53]. Two fuzzy sets *A* and *B* are equal (Eq. 3.8) if every element of the universe has the same membership degree in each of them; and *A* is a subset of *B* (Eq. 3.9), if every element of their discussion universe *U* has a lower membership degree in *A* than in *B*.

$$A = B \Leftrightarrow \forall u \in U : \mu_A(u) = \mu_B(u) \tag{3.8}$$

$$A \subseteq B \Leftrightarrow \forall u \in U : \mu_A(u) \leq \mu_B(u) \tag{3.9}$$

Applying the complement, intersection and union operations to fuzzy sets requires their generalization [29]. The conceptual consideration of the complement leads to the definition of the *c* operator defined in the set of Eq. 3.10, what as a general form comprehends the common form used with classical sets too, which applied to fuzzy sets is $\mu_{A'}(x) = 1 - \mu_A(x)$.

$$\begin{aligned} &\forall u \in U : \mu_{A'}(u) = \mathrm{c}\left(\mu_A(u)\right) \\ &\text{where}: \begin{cases} \mathrm{c}(0) = 1 \\ \mathrm{c}(\mathrm{c}(a)) = a \\ \mathrm{c}(1) = \mathrm{c}(\mathrm{c}(0)) = 0 \end{cases} \end{aligned} \tag{3.10}$$

In a more linguistic approach the intersection, union and complement operations correspond to the logic operators ‘and’, ‘or’ and ‘not’, which have a well defined semantic. In a more general form, intersection and union are represented by the *T-norm* and the *S-norm*, respectively.

$$\begin{aligned} &\forall u \in U : \mu_{A \cap B}(u) = \mathrm{T}(\mu_A(u), \mu_B(u)) = \mu_A(u) \hat{*} \mu_B(u) \\ &\text{where} \begin{cases} a \hat{*} b = b \hat{*} a \\ (a \hat{*} b) \hat{*} c = a \hat{*} (b \hat{*} c) \\ a \leq c \text{ and } b \leq d \text{ implies } a \hat{*} b \leq c \hat{*} d \\ a \hat{*} 1 = a \end{cases} \end{aligned} \tag{3.11}$$

A Triangular norm or T-norm $\hat{*}$, denotes a class of functions T: [0,1]×[0,1]→[0,1] that can represent the intersection operations, satisfying some criteria, presented in Eq. 3.11. In the same way, triangular co-norm or S-norm $\check{*}$, denotes a class of functions S:[0,1]×[0,1]→[0,1], that can represent the union operation, as shown in Eq. 3.12.

$$\begin{aligned} &\forall u \in U : \mu_{A \cup B}(u) = \mathrm{S}(\mu_A(u), \mu_B(u)) = \mu_A(u) \check{*} \mu_B(u) \\ &\text{where} \begin{cases} a \check{*} b = b \check{*} a \\ (a \check{*} b) \check{*} c = a \check{*} (b \check{*} c) \\ a \leq c \text{ and } b \leq d \text{ implies } a \check{*} b \leq c \check{*} d \\ a \check{*} 0 = a \end{cases} \end{aligned} \tag{3.12}$$

3.2.4 Triangular Norms and Co-norms

Many functions fulfill the requirements stated in Eqs. 3.11 and 3.12, therefore different ones can be used to represent union and intersection. However, there is a general relation between

them, presented in Eq. 3.13, which shows that such functions should be considered in pairs, as conjugates, that means, in a statement containing ‘and’ and ‘or’ connectors, when a T-norm is used the conjugate S-norm should be used as well.

$$a \hat{*} b = 1 - \left((1-a) \check{*} (1-b)\right) \tag{3.13}$$

Any function can be a T or S norm, since the conditions of Eqs. 3.10 or 3.11 are respectively respected, what possibilities to state an infinity number of such norms. Some T-norms are presented in Table 3.1.

Table 3.1: Examples of T-norms

Name	*Definition*
Minimum	$a \hat{*} b = \min(a, b)$
Algebraic Product	$a \hat{*} b = a \times b$
Bounded Product	$a \hat{*} b = \max((a + b\text{-}1), 0)$
Drastic Product	$a \hat{*} b = \begin{cases} a, \text{if } b = 1 \\ b, \text{if } a = 1 \\ 0, \text{if } a < 1 \text{ and } b < 1 \end{cases}$

In the same way, different S-norms can be stated, and some of them are shown in Table 3.2.

Table 3.2: Examples of S-norms

Name	*Definition*
Maximum	$a \check{*} b = \max(a, b)$
Probabilistic Sum	$a \check{*} b = a + b - a \times b$
Bounded Sum	$a \check{*} b = \min((a + b), 1)$
Drastic Sum	$a \check{*} b = \begin{cases} a, \text{if } b = 0 \\ b, \text{if } a = 0 \\ 1, \text{if } a > 0 \text{ and } b > 0 \end{cases}$

In Figures 3.1 and 3.2 the behavior of the T-norms of Table 3.1 are shown, as the behavior of the S-norms of Table 3.2 are in Figures 3.3 and 3.4 as well.

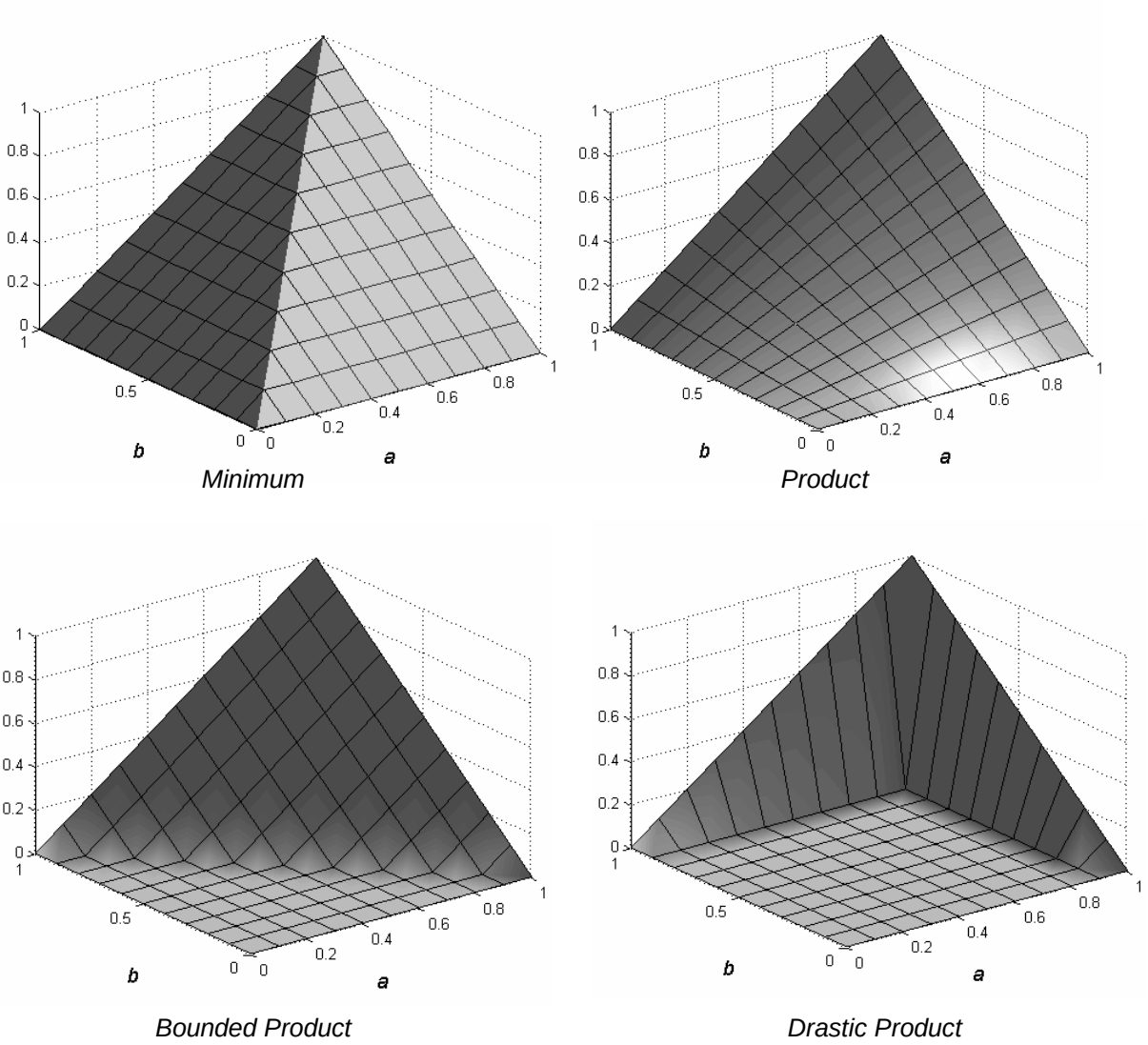

Fig. 3.1: Different T-norms

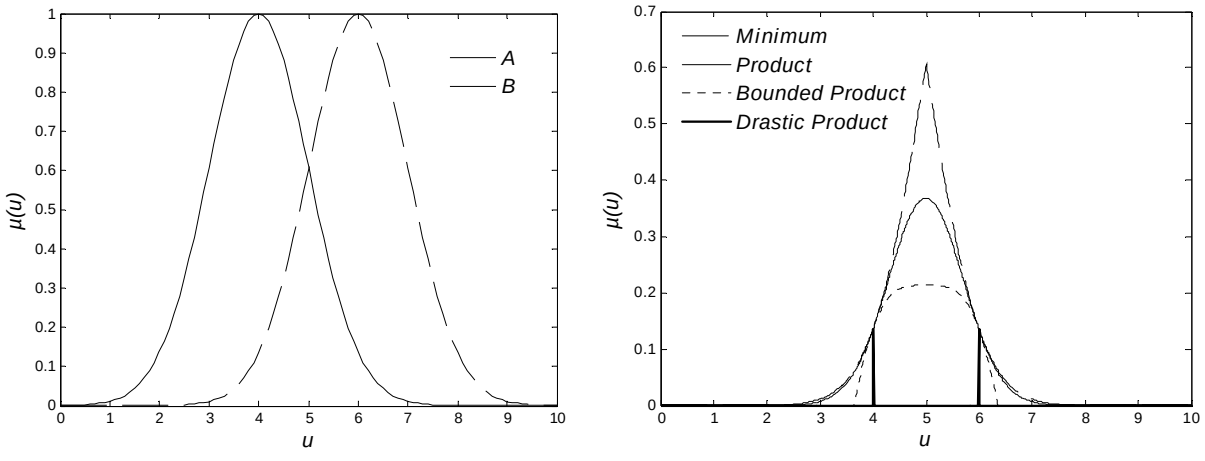

Fig.3.2: Application of different T-norms to fuzzy sets *A* and *B*

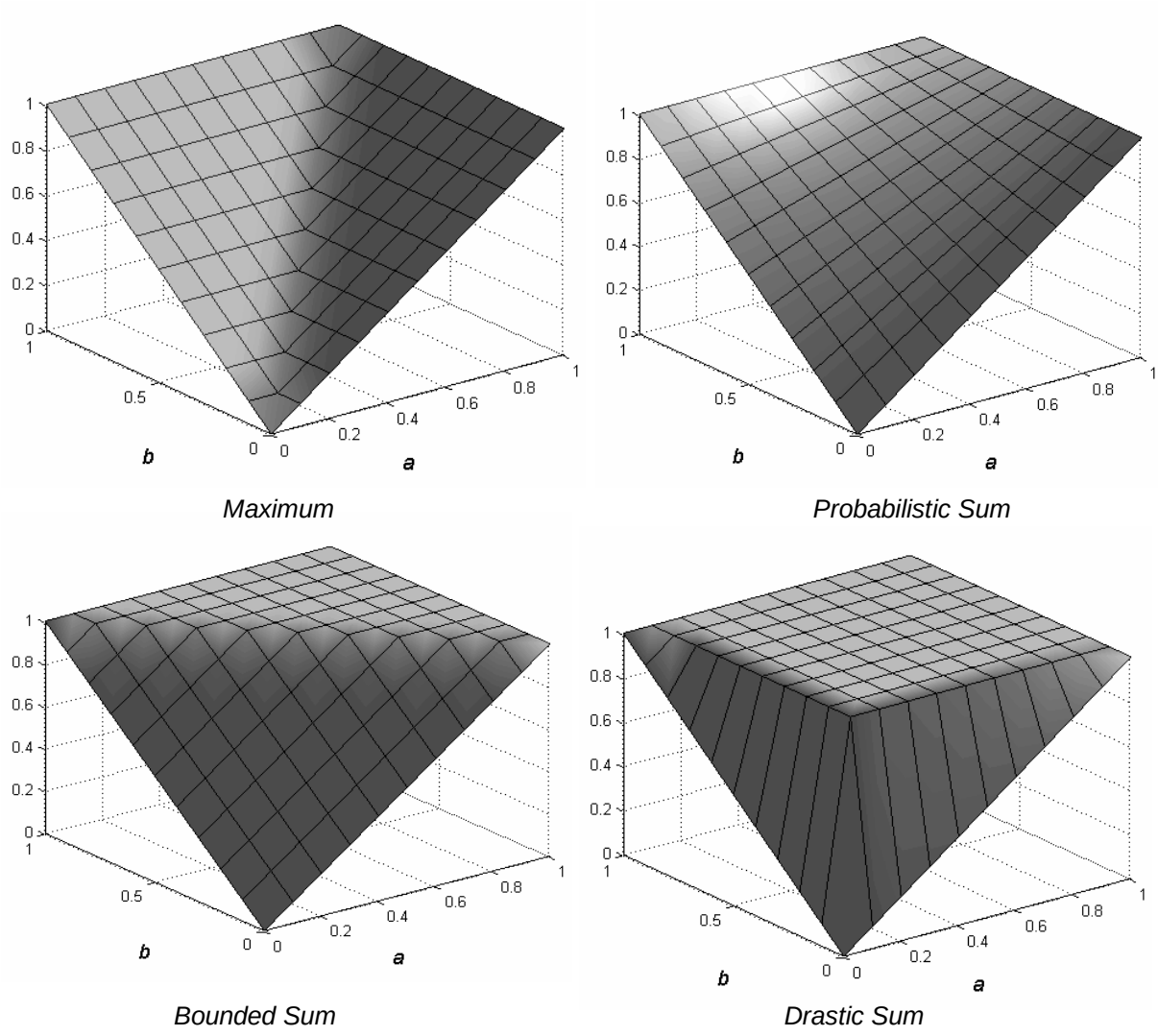

Fig. 3.3: Different S-norms

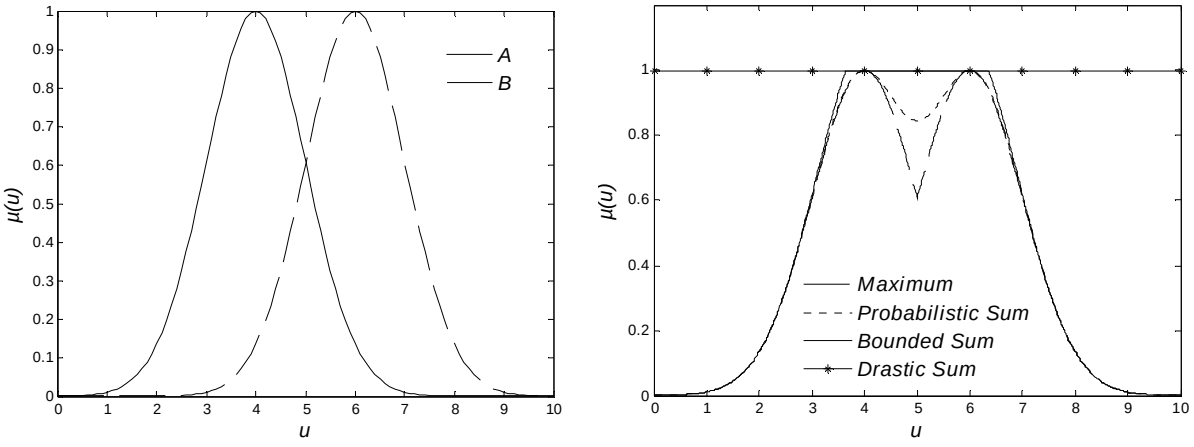

Fig.3.4: Application of different S-norms to fuzzy sets *A* and *B*

Naturally, the employment of different norms will lead to different results. In Fig. 3.3 the intersections of sets *A* and *B* is calculated using different norms. Each one gives a different result, where the 'lowest' is sad to be the most conservative of the group. The use of one or another depends on the interpretation of the problem, if a more or less conservative result is desired.

3.3 Fuzzy Relations

In the classical conception, a relation can be considered as a set of tuples, where a tuple is an ordered pair. In the same way, a fuzzy relation *R* is a fuzzy set of tuples, i.e., each tuple has a membership degree $\mu_R(u_1, u_2, \ldots, u_n)$: $U_1 \times U_2 \times \ldots \times U_n \rightarrow [0,1]$, where U_i are the discussion universe of the corresponding u_i. I particular, unary fuzzy relations are fuzzy sets with one dimensional membership functions. As a direct extension [126], considering *m* different relations R_j, the intersection and union operation can be stated as in Eqs. 3.14 and 3.15, respectively.

$$\begin{aligned} &\forall(u_1,\ldots,u_n) \in U_1 \times \ldots \times U_n : \\ &\mu_{R_1 \cap \ldots \cap R_m}(u_1,\ldots,u_n) = \mu_{R_1}(u_1,\ldots,u_n) \hat{*} \ldots \hat{*} \mu_{R_m}(u_1,\ldots,u_n) \end{aligned} \tag{3.14}$$

$$\begin{aligned} &\forall(u_1,\ldots,u_n) \in U_1 \times \ldots \times U_n : \\ &\mu_{R_1 \cup \ldots \cup R_m}(u_1,\ldots,u_n) = \mu_{R_1}(u_1,\ldots,u_n) \check{*} \ldots \check{*} \mu_{R_m}(u_1,\ldots,u_n) \end{aligned} \tag{3.15}$$

Two very important operations on normally used on fuzzy relations are *projection* and *cylindrical extension*. The projection brings the relation to a lower dimension, e.g., a ternary relation back to a binary relation or a binary relation to a fuzzy set, or a fuzzy set to a single crisp value. To project the relation *R* on $U^{proj} = U_1 \times \ldots \times U_{k-1} \times U_{k+1} \times \ldots \times U_m$, eliminating the component *k* (which has *q* elements) the supremum is used, as shown in Eq. 3.16.

$$\mu_{R_{U^{proj}}}(u_1,\ldots,u_{k-1},u_{k+1},\ldots,u_m) = \sup_{u_{k_1},\ldots u_{k_q}} (\mu_R(u_1,\ldots,u_m)) \tag{3.16}$$

The cylindrical extension is more or less the opposite of the projection. It extends fuzzy sets to fuzzy binary relations, fuzzy binary relations to fuzzy ternary relations, etc. To extend the already defined relation *R* to $U^{ext} = U_1 \times \ldots \times U_m \times U_{m+1}$, follows as in Eq. 3.17.

$$\mu_{R_{U^{ext}}}(u_1,\ldots,u_m,u_{m+1}) = \mu_R(u_1,\ldots,u_m) \tag{3.17}$$

The combination of fuzzy sets and fuzzy relation with the aid of cylindrical extension and projection is called composition.

One important notion in fuzzy set theory is the *Extension Principle*. The extension principle provides a general method for combining non-fuzzy and fuzzy concepts of all kinds, e.g., for combining fuzzy sets and relations, but also for the operation of a mathematical function on fuzzy sets. Fuzzy sets can also be interpreted as fuzzy numbers, i.e., a central value with an inherent uncertainty. In this case, one can use the extension principle to add or multiply these fuzzy numbers. Consider the fuzzy relation R defined in last section, and a function $\mathbf{f}$: $U_1 \times \ldots \times U_n \rightarrow V_1 \times \ldots \times V_m$, then a new relation S over the image of f is defined as in Eq. 3.18.

$$\begin{aligned} (v_1,\ldots,v_m) &= \mathbf{f}(u_1,\ldots,u_n), \quad v_i \in V_i \\ \mu_S(v_1,\ldots,v_m) &= \sup_{u|v=\mathbf{f}(v)} (\mu_R(u_1,\ldots,u_n)) \end{aligned} \tag{3.18}$$

As shown, since the function can be inverted or at least an inverse relation mapping v onto u can be found, any function can be extended from the domain of real numbers to the domain of fuzzy numbers.

3.4 Approximate Reasoning

Such reasoning covers a variety of inference rules whose premises contain fuzzy propositions. Inference in approximate reasoning is in contrast to inference in classical logic - in the former, the consequence of a given set of fuzzy propositions depends on the meaning attached to these fuzzy propositions. Then, inference in approximate reasoning is computation with the fuzzy sets which represent the meaning the propositions. Therefore, the proposition represents a natural language form, while the corresponding fuzzy set is its mathematical representation.

The 'meaning' (interpretation) of an expression is directly related to the defining predicative or characteristic property of the fuzzy set involved in the expression. The membership function of such a fuzzy set represents its meaning and is defined on the normalized physical domain of the discussion universe of the proposition, which can be something so simple as 'u is A'. In this proposition, $u \in U$, the universe where the membership function μ_A is defined to represent the meaning of the proposition. The particle 'is' actually means 'has the property of being'; hence, μ_A quantifies how much this property is satisfied.

The fundamental knowledge representation unit in approximate reasoning is the notion of a *linguistic variable*, which means a variable whose values are words or sentences in a natural

or artificial language. A variable assignment takes a symbol to denote a physical variable and returns a physical value from the membership functions after the reasoning.

Based on the notion of fuzzy propositions and linguistic connectives such as 'and', 'or' and 'not' one can form more complex fuzzy propositions called compound fuzzy propositions, whose meaning are given by interpreting the connectives 'and', 'or' and 'not' as intersection, union and complement, respectively. Another fundamental connector is the so-called 'if-then', which makes possible to construct conditional fuzzy statements. A fuzzy conditional or a fuzzy if-then rule is symbolically expressed as 'if < *fuzzy proposition* > then < *fuzzy proposition* >', where each fuzzy proposition is either a simple or a compound one. When the rule antecedent or the rule consequent is a compound fuzzy proposition then first the membership function corresponding to each such compound proposition is determined.

3.4.1 Inference Rules

In approximate reasoning, two inference rules are of major importance: the compositional rule of inference and the generalized *modus ponens*. The first rule uses a fuzzy relation to represent explicitly the connection between two fuzzy propositions; the second one uses an 'if-then' rule that implicitly represents a fuzzy relation. The generalized *modus ponens* has the symbolic inference scheme from Eq. 3.18:

$$\begin{aligned} & u \text{ is } A', \\ & \text{if } u \text{ is } A \text{ then } v \text{ is } B, \\ \therefore\ & v \text{ is } B'. \end{aligned} \tag{3.18}$$

where u and v denote symbolic names for objects, and A, B, A' and B' denote object properties. Due to the representation of the meaning of the properties in terms of fuzzy sets, a conclusion can be derived even when the input is A' instead of A. The compositional rule of inference can be considered as a special case of the generalized *modus ponens*, with its general symbolic form as in Eq.3.19, where u R v reads as 'u is in relation R to v' and its meaning is represented as a fuzzy relation μ_R. Hence, instead of the 'if-then' rule, there is a fuzzy relation R:

$$\begin{aligned} & u \text{ is } A', \\ & u \text{ R } v, \\ \therefore\ & v \text{ is } B'. \end{aligned} \tag{3.19}$$

4 Data Fusion

For an autonomous system it is important to be able to sense its own environment, but usually the system cannot rely on a single sensor to provide sufficient information because in general, sensor measurements contain noise and can be erroneous. Measurements of a single sensor can be incomplete but different types of sensors can measure different properties of the world, then data from multiple sensors must be fused. To this end, some of the measurement properties have to be in common, otherwise the fusion is not possible. Moreover, autonomous systems will often operate in a dynamic world with moving objects, which means that fusion has also to take place over time.

Because information from sensors is incomplete and uncertain, it is essential that the system is able to fuse redundant information from multiple sensors. The term redundant means that multiple sensors should measure (partly) the same information to reduce the uncertainty in this information and to solve the correspondence problem in incomplete information. Sensor data fusion is basically the fusion of information from different sensory sources in order to obtain more meaningful information of the surroundings.

4.1 Single Sensor and Multiple Sensors Fusion

Data coming from a single sensor can be fused by taking the measurements at successive time intervals. In this way, the successive measurements are fused to obtain a more accurate description of the environment than with a single measurement, it is tried to enhance the certain about the measured value, what is called ‘fusion-in-time’. This method can be easily applied when some parametric description of the environment is available. In that case, the measurements are used to successively update the estimation of the parameters of the world representation.

Figure 4.1 shows an example of fusion, where different measurements done by a single sensor are used to estimate a parameter value. Three measures are taken, each one comprehending some uncertainty, which is represented by a certainty function that expresses the possibility of a value to be the real parameter value, given a measure. The three measures are fused using a *minimum* operator and normalization, leading to a final value with smaller uncertainty. This means that an enhancement of the certain about the real parameter value was achieved. Therefore, a better estimate is obtained because the random error component can be reduced. Systematic errors cannot be reduced by this kind of fusion.

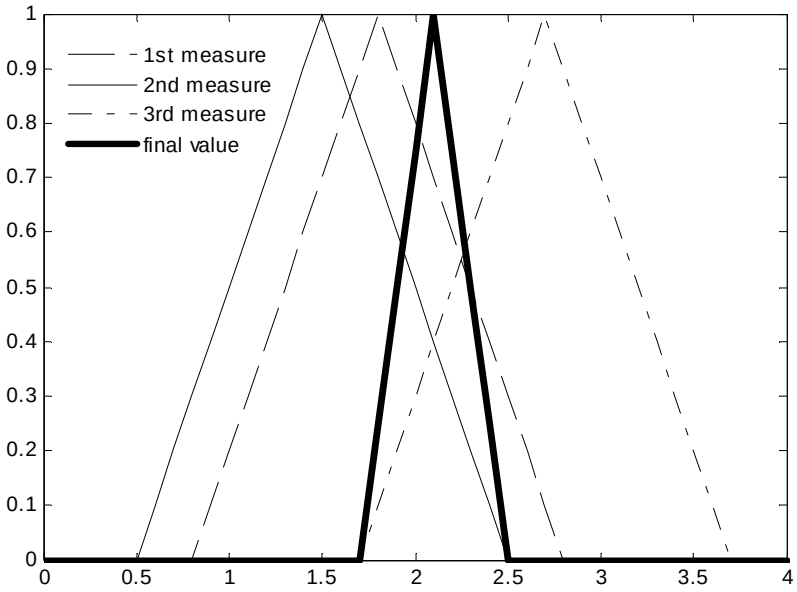

Fig. 4.1: Enhancement of certain by fusion

Seldom can a single sensor provide sufficient information for the reasoning component in the autonomous system. Hence, the system must fuse information from multiple sensors to obtain more complete and more accurate information about the world. In general, three types of sensor data fusion can be distinguished:

- *Complementary fusion*: fusion of several disparate sensors, which only give partial information of the environment. In this case many sensors are used to cover the whole environment, each one covering only a part of it, as in Fig. 4.2. This type of fusion resolves incompleteness of sensor data.

- *Competitive fusion*: fusion of uncertain sensor data from several sources (transducers). This type of fusion is predominantly aimed at reducing the effect of uncertainty in erroneous measurements. Since the estimation of parameters is prone to errors, more accurate estimates can be obtained by fusing multiple measurements of the same parameter, as in Fig. 4.3. Fusion can either be performed on several measurements from different sensors at the same time or on several measurements from the same sensor at different times. The already mentioned ‘fusion-in-time’ is a particular case of competitive fusion.

- *Cooperative fusion*: fusion of different sensors, where one sensor relies on the observations of another sensor to make its own observations. It diminishes uncertainty, measurements errors, as well as incompleteness, but it is actually not very common and will not be subject of a deep discussion here.

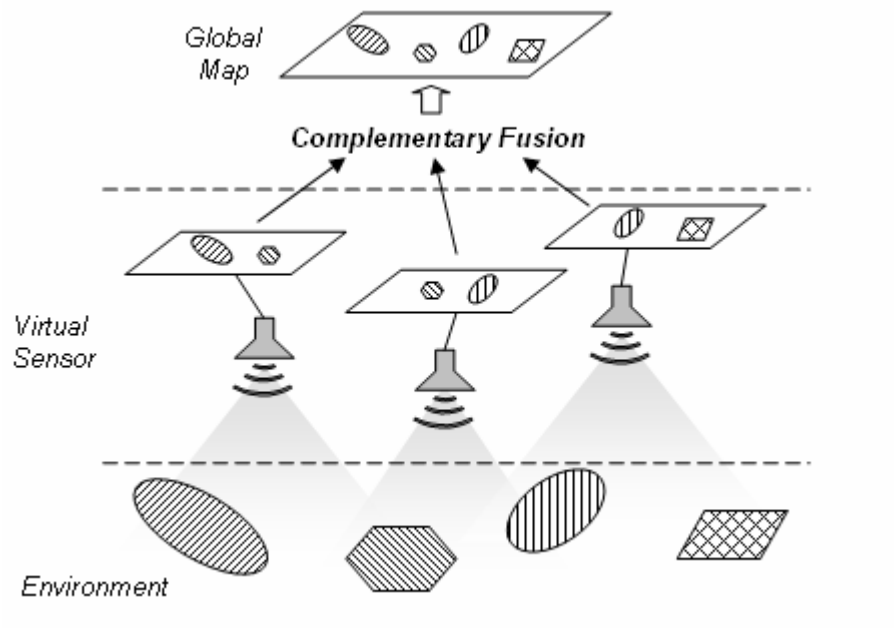

Fig. 4.2: Complementary Data Fusion

Such classification is in general just theoretical, in practical applications it is not always possible to distinguish if a fusion process is competitive or cooperative, or to separate such components because they are intrinsically associated through the measurement process. However, the presentation of this classification is worth because it shows the approach to be used when having a problem of incompleteness or uncertainty.

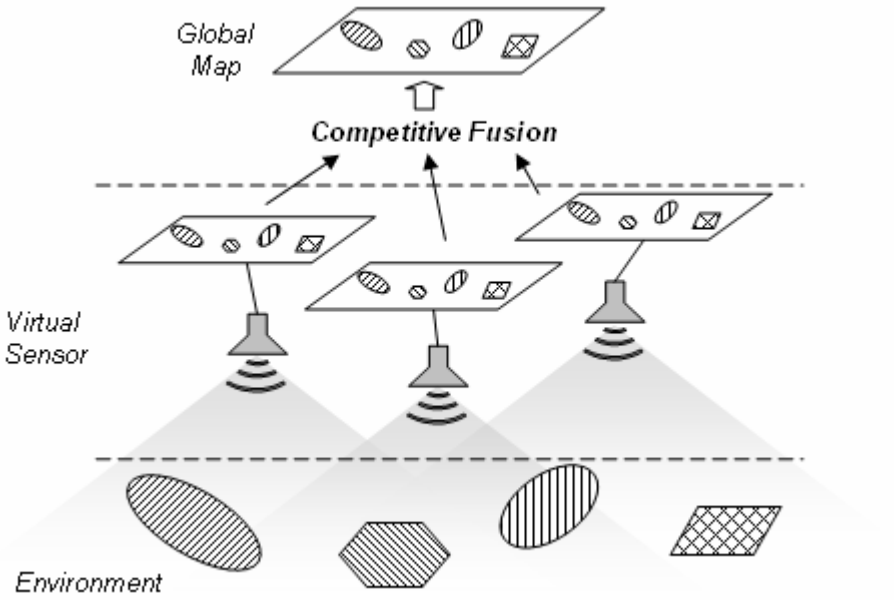

Fig. 4.3: Competitive Data Fusion

4.2 Data Fusion System

The most important issue in a data fusion system is the choice of a suitable representation. A generic sensor data fusion system should be able to fuse data from sensors of many different modalities; therefore, for all data from these disparate sensors to be fused, it is necessary that they follow the same standard. In other words, all sensors should first convert their measurements to a common representation before the actual fusion is performed. In Figures 4.2 and 4.3, it can be noted that the main characteristics of the employed architecture are:

- data from each measuring device are first converted to an internal representation;
- this internal representation is common to all sensors;
- the actual fusion of data is performed in this internal representation.

In this architecture, sensors are treated as what was previously called virtual sensors: the sensor performs both the actual measuring and the conversion to an internal representation. Although this architecture seems to be very natural in the context of virtual sensors, some systems perform the actual fusion before the conversion to the internal model. This means that one sensor relies on another sensor's observations to make a conversion to an internal model, what was previously called cooperative fusion. In short, cooperative fusion is performed within one single virtual sensor (what means more than one transducer in the same virtual device), while competitive and complementary fusions are performed in the internal representation.

If higher-level fusion methods are to perform competitive and complementary fusion successfully, it is important that the sensors not only give an estimate of the parameter in the internal representation but also a *certainty value* for this estimate. This certainty value expresses how certain the sensor is about its estimation of the parameter and can be used in the subsequent fusion process. A simple example is the conjugated employment of ultrasonic and infrared range finders to measure distances and angles to obstacles with respect to the current robot position and orientation. An acoustic sensor can estimate the distance to an object quite accurately while it is less certain about the angle at which the object is found; in contrast to this, an infrared sensor could quite accurately determine the angle, while it would be less certain about the distance. If only the estimates given by each sensor would be available, it would not be possible to use these sensor characteristics, and the estimated parameter could only be straightforward averaged. However, if also certainty values were available, a weighted average of the estimated parameters could be computed, giving more weight to the distance estimate of the acoustic sensor and to the angle estimate of the infrared sensor.

In a general way, the parameters in the internal model are considered in form of *propositions*, e.g., 'at angle *a* from robot's current configuration there is an obstacle present at distance *r*'. In this proposition, *a* and *r* are the parameters of interest and the whole proposition is the internal common representation to which the measurements are converted and to which certainty values are attributed. As the conversion to a unique internal representation and the evaluation of certainty are executed, a higher-level fusion method can be applied, where one of the most widely used approaches is the Bayesian one, which will be presented together with some other non-Bayesian approaches in the following sections.

4.2.1 Bayesian approach

The representation of certainty values of a proposition φ, given a sensor measurement d, is done by functions called the *inverse sensor model*, which is derived from the sensor properties. In the Bayesian approach, such a model is represented by the conditional probability density function $p(\varphi \mid d)$, which is calculated by application of the Bayes rule, shown in Eq. 4.1:

$$p(\varphi \mid d) = \frac{p(d \mid \varphi)}{p(d)} \cdot p(\varphi) \tag{4.1}$$

Through Eq. 4.1, the measurement is converted by giving certainty value of the proposition φ regarding the measured value d. Consider now two different sensors *1* and *2*, which make respectively measurements d_1 and d_2 of the same object. As they are different sensors, they have different inverse sensor models p_1 and p_2 to convert their measurements, what reflects the fact that the sensors have different error characteristics as can be seen in Fig. 4.4.

As the two probability density functions are available, fusion can be performed by taking their product. It is assumed that the sensor measurements are independent, i.e., $p(d_1 \mid d_2) = p(d_1)$ and $p(d_1 \mid \varphi, d_2) = p(d_1 \mid \varphi)$, then the joint probability density function $p_{joint}(\varphi \mid d_1, d_2)$ in Eq. 4.2 is obtained.

$$p_{joint}(\varphi \mid d_1, d_2) = \frac{p(\varphi \mid d_1) \cdot p(\varphi \mid d_2)}{p(\varphi)} \tag{4.2}$$

Equation 4.2 shows how the competitive fusion is done; that is, when both sensors measure the same obstacle. When the sensors measure different obstacles complementary fusion is to be employed, that in the Bayesian approach is simply performed by adding the different probability density functions.

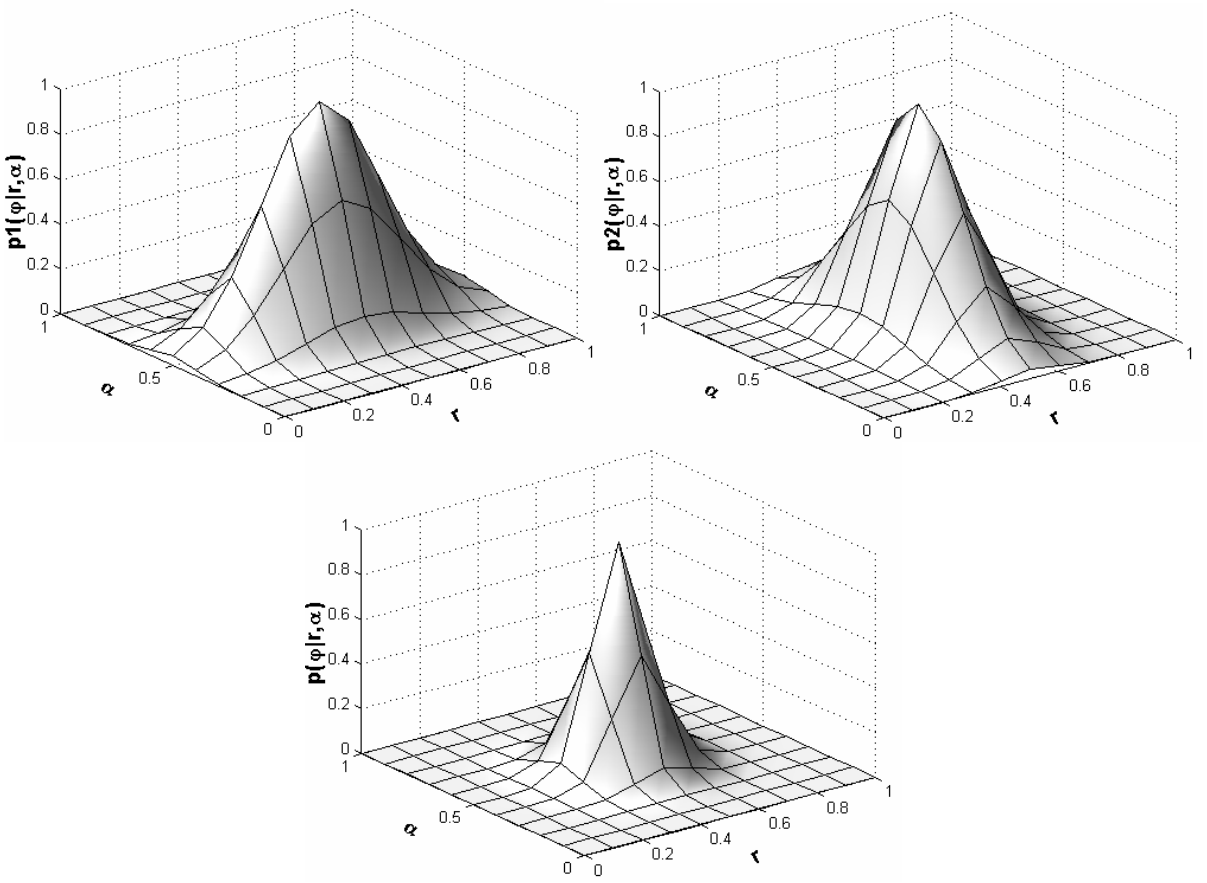

Figure 4.4: Two different sensor outputs and their fusion.

The *Occupancy Grid* is one of the most used techniques for robot navigation [31] and is originally based on the Bayesian approach for the data fusion. It is assumed that the workspace is quantized, described by a 2D grid, and that to navigate through this workspace it is necessary to know whether the workspace is occupied by obstacles or not. Two propositions are stated: 'the cell is occupied' (*Occ*) and 'the cell is empty' (*Emp*), which are associated by probabilities to each cell C_{ij}. Then, the probability that a cell is occupied is $p(C_{ij} = Occ)$; moreover, in the Bayesian approach it is assumed that $p(C_{ij} = Occ) + p(C_{ij} = Emp) = 1$, what together to the assumption that cell status are independent variables leads to assume that $p(C_{ij} = Occ) = 0.5$ when there is no information. After a sensor reading d the occupancy grid is updated using the Bayes theorem as in Eq. 4.3:

$$p_{joint}\left(C_{ij} = Occ \mid d\right) = \frac{p\left(d \mid C_{ij} = Occ\right)}{p(d)} \cdot p\left(C_{ij} = Occ\right) \tag{4.3}$$

The above equation indicates that to update the probabilities from a measurement d it is necessary to know $p(d \mid C_{ij} = Occ)$ and $p(d)$. To estimate these probabilities all possible world configurations with $C_{ij} = Occ$ have to be applied to the sensor model, in order to find the

probability of having a measurement *d*. A question arises: what are the possible worlds for the robot? In structured terrains it is still possible to make a choice about what representative environments the robot will operate in, but not in general. Some other more general drawbacks of this approach can be mentioned:

- Since $p(C_{ij} = Occ) = p(Cij = Emp) = 0.5$ is assumed when no information is present, there is a lack of confidence. The situation is different when there is no information present or when in half of the cases the outcome is empty and in half of the cases the outcome is occupied.
- All conditional probabilities must be specified. With a poor sensor model, these conditional probabilities have a large influence on the final results.
- Presence of redundant readings, that conflicts with assumptions about independence.
- Usually a Gaussian sensor model is used, what means to use a Normal distribution as inverse sensor model but specular reflections do occur frequently and are not taken care of in this case. Figure 4.5 shows a typical case of specular reflection [14]: the measured distance is half the length of the whole traced line, but the actual distance to be measured corresponds to just the first segment.

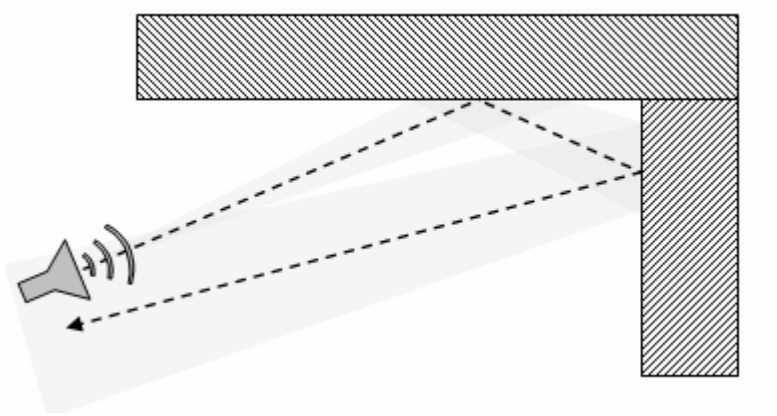

Fig. 4.5: Specular reflection

4.2.2 Dempster-Shafer Approach

In this approach a virtual sensor converts its measurements by giving a set of possible values for the parameters. Fusion is performed on virtual sensor data through intersection, what diminishes the widths of these sets, as shown in Fig. 4.6. As it was already mentioned, a problem with the Bayesian approach is that the distribution of probabilities may not be known. In this case, a possibilistic approach is preferable to a probabilistic one.

A finite set $\Phi = \{ E_1, E_2, ..., E_n\}$of mutually exclusive events E_i must be stated, so that these events are able to describe the environment characteristics that are interesting for the robot. In the Dempster-Shafer theory, propositions are composed by the events; hence each proposition may comprehend a disjunction, i.e., subsets of the set of possible events as well as a singleton, i.e., only an event itself. The certainty values *m* are then assigned to the propositions and must still sum to 1; as the disjunctions may have non-empty intersections, all possible subsets of the events set are considered, what means that certainty values are associated to each element *A* of the power set Λ of Φ, as in Eq. 4.4.

$$\Lambda = \{\varnothing, E_1, E_2, \ldots, \{E_1, E_2\}, \ldots, \{E_{n-1}, E_n\}, \ldots, \{E_1, \ldots, E_n\}\}$$
$$\sum_{A \subset \Lambda} m(A) = 1 \tag{4.4}$$

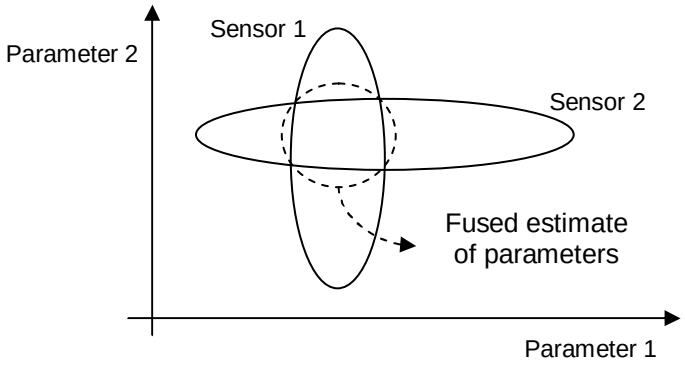

Fig. 4.6: Uncertainty sets of different sensors and their fusion

The subset $\{\varnothing\}$ represents the unknown state in which none of the events in the set occurs, what in many applications can never occur, in which case its certainty value is 0. Moreover, the total evidence attributed to some proposition and to its corresponding subset *A*, is the sum of all certainty values assigned to *A* and all the subsets of which *A* is a part. This is called the belief in *A*. The actual fusion of beliefs, that is, the fusion of converted sensor measurements is performed with Dempster's rule of combination: consider two sensor measurements *B* and *C* that result in certainty values m_1 and m_2 for a subset *A*, the combination is given by Eq. 4.5.

$$m_1 \oplus m_2(A) = \frac{\sum_{\forall B, C \in \Lambda : B \cap C = A} m_1(B) \cdot m_2(C)}{1 - \sum_{\forall B, C \in \Lambda : B \cap C = \varnothing} m_1(B) \cdot m_2(C)} \tag{4.5}$$

Basically the product of the certainty values of all subsets in which the intersection is exactly *A* is calculated and divided by 1 minus the product of the certainty values of the subsets that

do not intersect, what corresponds to divide the intersection of beliefs in the proposition generated by each measurement by the intersection of the beliefs (from each measurement) in its negation. The above formula forms the basis for sensor fusion in a possibilistic approach.

Here the technique of Occupancy Grids will be used again, now to exemplify the application of Dempster-Shafer approach to fuse data obtained with an ultrasonic sensor. The sensor model incorporates the available evidence for empty space (*Emp*) and for occupied space (*Occ*), hence the set of all possible events is $\Phi = \{Occ, Emp\}$ with the corresponding power set $\Lambda = \{\varnothing, Occ, Emp, \{Occ, Emp\}\}$.

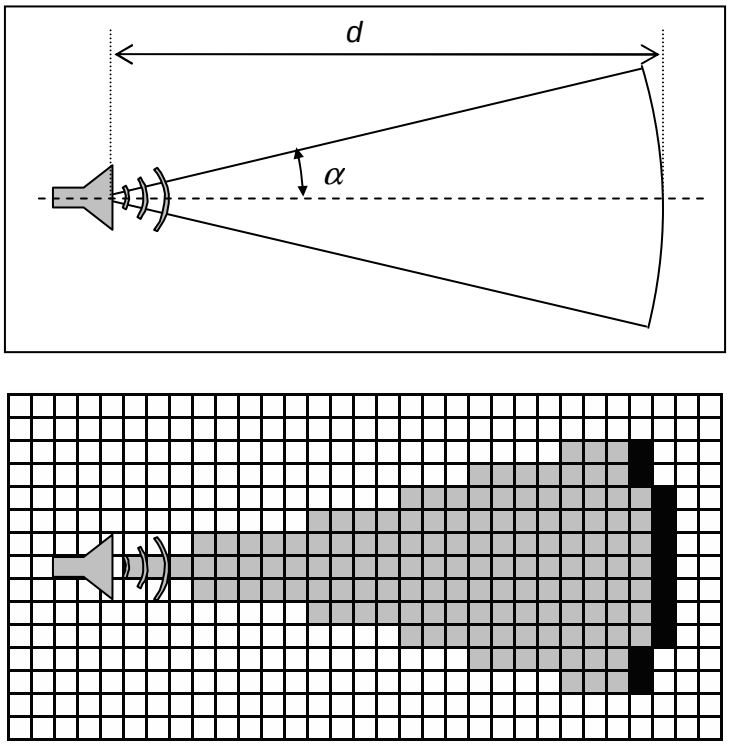

Fig 4.7: Modelling of an ultrasonic sensor by possibilistic approach.

In Fig 4.7 the evidence of existence of an occupied cell on the ultrasonic sensor's arc is sketched, whose length is $2d\alpha$ and which is supposed to cover n cells (the black area). That results in a certainty value of $1/n$ for the occupancy of each cell on the arc, but no information whether a cell on the arc is empty was obtained, as shown in Eq 4.6. In the same way, there is evidence of emptiness for the cells on the sector (grey area in Fig. 4.7), what is modelled by a certainty value ρ. In this case, no information whether a cell is occupied is available, as stated by Eq.4.7. The certainty values for cells outside the sector and arc are described in Eq. 4.8.

$$\left.\begin{array}{l} m_{ij}(Occ) = 1/n \\ m_{ij}(Emp) = 0 \end{array}\right\} \forall C_{ij} \in \text{arc} \tag{4.6}$$

$$\left.\begin{array}{l} m_{ij}(Occ)=0 \\ m_{ij}(Emp)=\rho \end{array}\right\} \forall C_{ij} \in \text{sector} \tag{4.7}$$

$$\left.\begin{array}{l} m_{ij}(Occ)=0 \\ m_{ij}(Emp)=0 \end{array}\right\} \forall C_{ij} \notin (\text{arc} \cup \text{sector}) \tag{4.8}$$

Now it is necessary to fuse the certainty values $m_S(Emp)$ and $m_S(Occ)$ resulting from the sensor measurements, with the certainty values $m_M(Emp)$ and $m_M(Occ)$ from the current map (grid) of the environment. In Eq. 4.9 the indices *i,j* for individual cells are dropped for clarity and the rule of combination is applied for each cell. It results somewhat complex because of the beliefs may be expressed for disjunctions; indeed, one of the main disadvantages of Dempster-Shafer theory is that its application leads to exponential complexity:

$$\begin{aligned} m_S \oplus m_M(Emp) = & \\ & \frac{m_S(Emp)\cdot m_M(Emp)+m_S(Emp)\cdot m_M(\{Emp,Occ\})+m_S(\{Emp,Occ\})\cdot m_M(Emp)}{1-(m_S(Emp)\cdot m_M(Occ)+m_S(Occ)\cdot m_M(Emp))} \\ m_S \oplus m_M(Occ) = & \\ & \frac{m_S(Occ)\cdot m_M(Occ)+m_S(Occ)\cdot m_M(\{Emp,Occ\})+m_S(\{Emp,Occ\})\cdot m_M(Occ)}{1-(m_S(Emp)\cdot m_M(Occ)+m_S(Occ)\cdot m_M(Emp))} \end{aligned} \tag{4.9}$$

Summarizing, the most pronounced features of the theory are:

- Possibility to assign beliefs to disjunctions of propositions without assigning beliefs to the individual propositions.
- Possibility to leave belief uncommitted, what is comparable to assigning a probability to a special ‘unknown’ proposition.

4.2.3 Fuzzy logic

Another possibilistic approach, which has attracted much attention, is the fuzzy logic approach. The method does not assign probabilities to propositions but rather assigns membership values to pre-defined sets [97]. The essential feature of fuzzy logic is that the boundaries of the sets are not crisp but fuzzy. Basically, the fuzzy-logic approach offers the same advantages as the Dempster-Shafer theory (they are both possibilistic approaches) whereas the fuzzy logic approach does not suffer from the exponential complexity [36].

Another attractive side to fuzzy logic is that the membership values can be assigned with intuitively comprehensible values. This makes the fuzzy logic approach especially applicable

to capturing knowledge from human experts, which is often hard to represent in precise numerical values. Conversely, it is also much easier for a human operator to comprehend the rules in a fuzzy logic system.

Again, the actual competitive and complementary fusion is performed with rules of combination of fuzzy values, which do not differ much, in effect, from the Bayesian rules of combination. Like Bayesian theory, in fuzzy logic theory also intersection and union are used; roughly speaking, the former is for competitive fusion and the latter for complementary fusion. A deeper discussion about this approach will be carried in next chapter.

4.2.4 Sensor fusion by learning

Obtaining a correct inverse sensor model for a given situation is a serious problem, because the models mentioned so far, both probabilistic and possibilistic, are based on ideal sensor models but reality is in general much more complex. Realistic responses can be learned using a neural network [4], in which a robot moves in an environment, where is known which cells are occupied and which are empty, and what the measurements of sensors are for each situation. A mapping between sensor measurement and occupancy of the cells surrounding the robot can be learned by training a neural network, with the sensor measurements as input and occupancy values of the cells as output, as in Fig. 4.8.

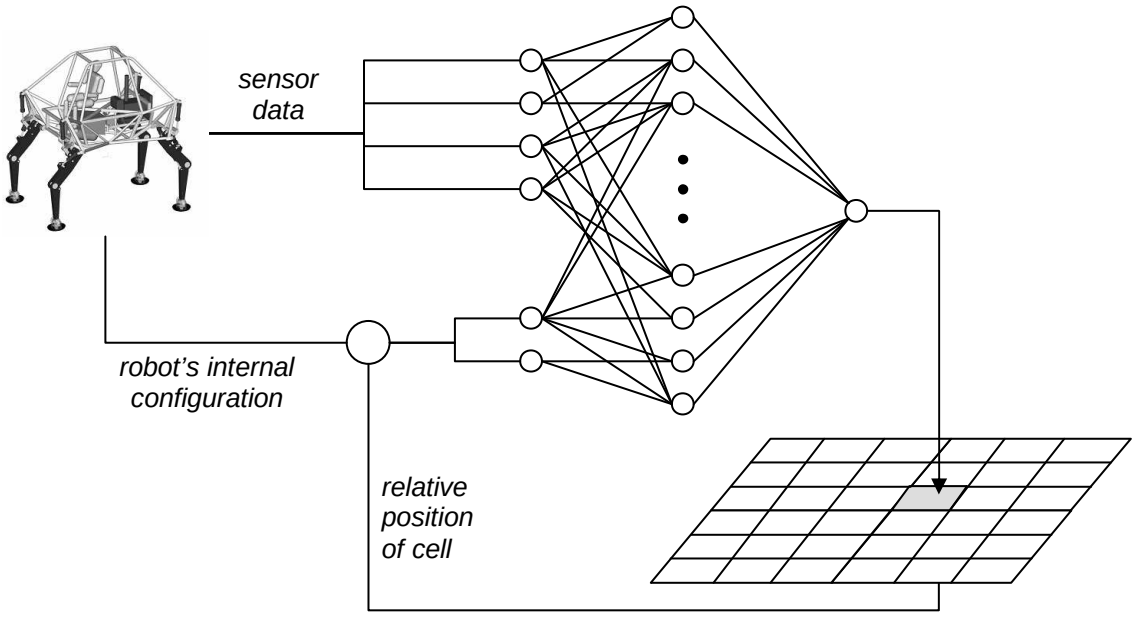

Fig. 4.8: Occupancy and Confidence Neural-Network

A neural network is composed of a combination of simple functions, with adjustable parameters. The tuning of such parameters through an optimization process in order to minimize the error between the real data (known) and the network output is called training.

Consider again the surrounding environment of the robot as a two dimensional grid and a given a set of sensor measurements $D = \{d_1, d_2, \ldots, d_n\}$. Since the grid is a local representation, if the robot stands on a position k, the a priori known occupancy grid $C(Occ)$ gives the local known occupancy state, while measurements D_k are done at this position. A mapping between $C(Occ)$ and D_k is stated by training a neural network, which will give estimates $C(Occ_{pred})$ of the occupancy state. This mapping can be much more reliable for some cells than for other ones (given the geometrical configuration if the environment), therefore the confidence in the prediction must be extracted from the expected error between the predicted and real occupancy values. Then, a second network is trained to the estimate confidence ξ_i in such values; however, the error will never become completely zero, because some cells in the environment may not be observable or be occluded, so information about them cannot be obtained.

After the training stage the networks are used to create a map M of the environment, fusing sensor interpreted values according to their confidence, as in Eq. 4.10, shown in Fig. 4.8.

$$M = \frac{\sum_k -\ln(\xi_k) \cdot C_k(Occ)}{\sum_k -\ln(\xi_k)} \tag{4.10}$$

As the neural networks are trained in a certain type of environment, the mapping would not be efficient in another one; therefore, a calibration is necessary whenever a new kind of environment is met, what means that previous knowledge about the environment is necessary.

4.3 Analysis aiming at ALDURO

The main problem considering the Bayesian approach is the calculation of the conditional probabilities involved in the process, i.e., to state a reliable inverse sensor model. As already mentioned, one possible way to do that is to choose a model and apply it to all possible configurations of the environment, calculating the probabilities directly. That demands the previous knowledge of the environment where the robot is going to operate or, if that is not possible, the consideration of all representative possible environments where the robot may operate. Depending on the size of the terrain where the robot is supposed to operate and on the refinement of the grid, the off-line calculation of probabilities can lead to an extremely

high amount of calculation. In the case of ALDURO, which is supposed to operate outsides in different kinds of terrain, this solution is not feasible.

In general what is really done is just to take a normal distribution as sensor model with constant parameters. However, that leads to a very poor estimation, being useful just in robot applications where the safety requirement is low. Because of its relative huge dimensions, high power and humans involved, it would be very dangerous if ALDURO undergoes a collision, therefore a reliable sensor system is necessary.

In the same way, the use of neural networks would not be an feasible solution, since a previous knowledge of the environment is necessary. On the other hand, as a possibilistic approach, the application of the Dempster-Shafer theory demands much less previous knowledge. Actually no knowledge about the environment is strictly necessary, and only a superficial one about the transducer itself. The drawback is the exponentially growing amount of calculation necessary for the fusion in this approach.

This exponential growth refers to the formation of the power set of the set of events and, as each element of the power set is taken into account for the fusion, each new event doubles the calculation. At first it may seem that just two events, occupied and empty, would have to be used, but that would be suited only for a robot able to walk just on relative plain surfaces. The primary employment of ALDURO is supposed to be in very rough terrains; hence a planar map would not be suitable to execute its tasks. In order to use a three dimensional grid, at least one more event would be necessary to be defined, what, considering the much bigger number of cells (because of the 3D grid), would lead to too high calculation costs.

As the possibilistic approach has shown intrinsic advantages of good estimate without need of previous knowledge, the development of a collision avoidance system based on a fuzzy logic fusion seems to be the most appropriated. Besides the already mentioned advantages, the ease of representing expert’s knowledge may become the task of stating an inverse model much easier; and parallel, the available reasoning tools in fuzzy logic enable the search of more simple fusion processes.

5 The Collision Avoidance System

As already mentioned, a robot needs information about the environment to be able to execute its mission. Thus sensors are necessary to extract this data, but really meaningful information has to be constructed from this simple sensor data to enable the navigation. For ALDURO this improved information is a map [85], obtained through a data fusion process, for which different approaches were discussed in the last chapter.

The fuzzy approach was considered the most appropriate one to design a collision avoidance system for ALDURO, and this is discussed in this chapter. First a suitable inverse sensor model is stated [86][87], considering the many requirements for an internal representation and the transducer behavior in the scope of the applied approach; then a fuzzy inference system – the TSK – is used to make the fusion. Finally a hybrid method [88], based on fuzzy logic too, is presented to carry on the robotic navigation, which will actually, based on the collision avoidance system, make corrections on the instructions given by the operator.

The first sections of this chapter are dedicated to the design of the whole interpretation process; in a later section a reasoning process for navigation will be developed, which is to be appended to the main control system of ALDURO. As the collision avoidance system is responsible for identifying obstacles and informing the main control system about them, the collision avoidance module does not comprehend the stage of actuation.

5.1 Inverse Sensor Model

As sensor measurements are prone to errors, a first interpretation of them is necessary, where the parameters are considered with their inherent uncertainty. Next they have to be fused, therefore it is necessary that they are represented in a common system. All that is performed by the inverse sensor model, a model that tries to estimate parameters of the real world from the sensor measurement. To perform this task, the common representation has to fulfil some requirements:

- It should be common to all sensors: measurements from different sensors must be converted to a common internal representation so that fusion can be performed.
- Only aspects of the robot’s environment necessary for the execution of the mission should be represented.

- Not only estimates for parameters in the internal representation should be generated but also certainty values to express confidence of a sensor in its own measurements.
- Higher-level fusion methods have to be applicable to the estimates of the parameters and the associated certainty values.
- The internal representation should be chosen such that the virtual sensor can easily convert the raw sensor measurements. More abstract representations are better suited for the reasoning component, while on the other hand the model should also be closely related to the actual values measured by the transducer.

Many different internal representation models meet these requirements, but some are worth to mention. One of the most common is the *geometric representation*, where the robot's external environment is modelled using a set of geometrical objects, whose parameters are estimated by the autonomous system. For example, a cubic-shaped obstacle is detected by the robot sensors and has its position and orientation estimated; then the proposition φ in this internal representation is φ = 'the position of the center of cube *i* is *u* and the orientation *v*'.

Closely related to this is the *partially geometric representation*, in which the geometric parameters of objects in a scene are to be estimated by the sensors [11]. But here, only those geometric objects to be handled directly by the autonomous system are represented, i.e., that which will cause a response from the robot (in this case, from the navigation system). This representation basically does not differ from the previous one; it merely represents only a subset of the geometric objects in the environment. Not so restrictive but more abstract is the use of a *set of classes* for internal representation, what leads to propositions like φ = 'object *i* in the image belongs to class *A*'. It is clear that such a language can only be used at higher levels of fusion; just one sensor can hardly distinguish between classes only from its single signal, then a higher-level fusion between different sensory systems may be required.

Thus, considering an internal representation for a collision-avoidance system, according to the second requirement stated at the beginning of this section, only those aspects relevant to collision free navigation should be represented, what means: representation of free space and occupied space in the robot's external environment. But it is important to note that ALDURO's legs perform truly spatial movements, in a workspace where the usable area to walk may be not flat, what has to be distinguished from an obstacle. Here arises the need for a three dimensional representation of the converted signal as of the fused information as well.

Not only should the occupancy of space be represented, but also the associated certainty values. One choice of representation which meets these requirements is the occupancy grid or certainty grid. This representation divides the robot's external environment in a number of cells and for each cell the certainty about its occupancy by an obstacle. But to obtain a 3D representation with good resolution, considering ALDURO's dimensions, an excessively high

number of cells would be necessary; therefore a better option is to search a 2-dimensional representation able to retain the spatial information; something like a topographic map.

5.1.1 Measurement Uncertainty

In Chapter 2 the kind of sensor to be used in this application was already chosen: the ultrasonic sensor. Different from other sensing media measuring the distance to a target, ultrasonic sensors have quite large emission beams, which makes it a priori impossible to know the exact direction. Actually, conjugated use of several ultrasonic sensors or analysis of the shape of the returning echo allows a good approximation of the direction to the detected object. However, such shape analysis requires high quality analogue sensors and a posterior, complex and time-consuming data processing. Besides, conjugated use of sensors requires the possibility to operate receivers and senders separately, what is not always possible, especially with cheap range finder units, where the whole detection is treated as a single operation.

Ultrasonic sensors are sensitive to environmental changes too. Such changes affect the sound velocity and consequently the time of flight of the echo, what can be noted in Eq. 5.1, considering the expression for wave propagation speed in a ideal gas as a function of the adiabatic constant γ, the gas constant $\mathcal{R}$, the gas molecular mass $\mathcal{M}$ and the absolute temperature T (in Kelvin):

$$s = \sqrt{\gamma \cdot \mathcal{R} \cdot T / \mathcal{M}} \tag{5.1}$$

Thus the speed of sound s is dependent on temperature too. Using r_a as the actual range, r_m as the measured range, T_a as the actual temperature and T_c as the calibration temperature, a simple formula to correct errors is derived in Eq. 5.2:

$$r_a = r_m \cdot \sqrt{T_a / T_c} \tag{5.2}$$

ALDURO is expected to operate outdoors, probably submitted to large temperature variations. Therefore, with T_c = 21°C and T_a = 42°C (something extreme), the variation in range would be about 3.5%. The ultrasonic range finder used in this application, as will be detailed in next chapter, has a maximal range of 6m, what in the presence of the above mentioned temperature variation would cause a systematic error of up to 21cm. In conjunction with a temperature sensor this error could be easily eliminated, in a cooperative fusion; but here the worst situation will be considered, and this value, which is still acceptable, will be considered as an uncertainty.

The low precision in these sensors refers basically to the poor angular resolution, since the nominal axial resolution is high enough for this application (circa 1%). Fig. 5.1 shows a beam

chart of the used range finder unit, with its gain as function of the angle between the sensor emission/reception axis and the echo returning direction. The chart can be used to find the sensor workspace; first consider that the emitted ultrasonic pulse has a power *P*, as sound propagates in spherical waves, the intensity of the pulse at a distance *l*, could described by the relation $I_l = P/(4\pi l^2)$.

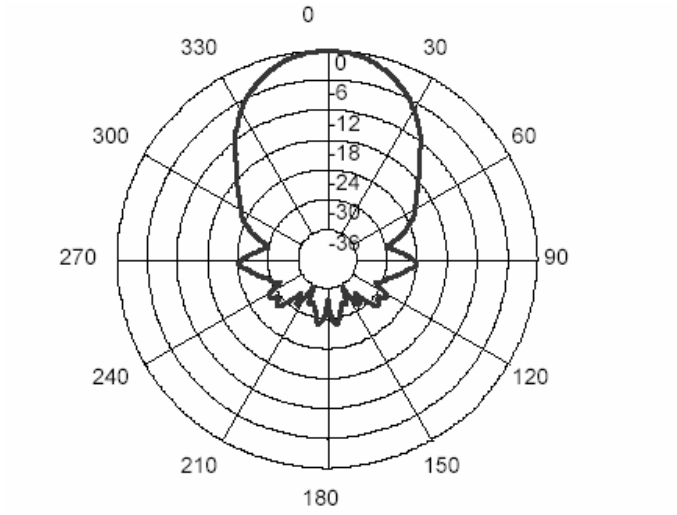

Fig. 5.1: Beam chart of the receiver (angles in degrees and attenuation in db)

To complete a measurement the pulse has to travel twice the distance to the obstacle, given a measured value *d*, the signal actually has traveled 2*d*. At the reflection point part of the signal is always refracted and/or absorbed, so an attenuation *k* due to these phenomena has to be considered. A second attenuation occurs when the signal is read by the transducer, and it is given by the receiver gain function *G*, plotted in the beam chart from Fig. 5.1, leading to Eq. 5.3, where α is the angle between two lines: the reception axis and the line from the receiver to the reflection point.

$$I_d = \frac{G(\alpha) \cdot k \cdot P}{16 \cdot \pi \cdot d^2} \tag{5.3}$$

For reception there is a threshold, a minimal intensity value I_{min} measurable by the receiver. Consider now a returning echo with the minimal intensity: since the intensity is minimal, the distance could be maximal with ideal reflection ($k = 1$). Figure 5.1 shows that the nominal maximal range *L* will occur at angle $\alpha = 0°$, and that $G(0°) = 0db = 1$, what leads to $I_{min} = P/(16.\pi.L^2)$.

Now, to state the contour of the sensor's workspace, it is necessary to know the maximal coverable range $r_{max,\alpha}$, for each value of the returning angle α. So, an ideal reflection should be considered again, with a returning echo of minimal intensity, what applied to Eq. 5.3 gives:

$$I_{\min} = \frac{G(\alpha)\cdot P}{16\cdot\pi\cdot r_{\max,\alpha}^2} \Rightarrow r_{\max,\alpha}^2 = \frac{G(\alpha)\cdot P}{16\cdot\pi\cdot I_{\min}} = L^2\cdot G(\alpha) \Rightarrow r_{\max,\alpha} = L\cdot\sqrt{G(\alpha)} \tag{5.4}$$

Actually, the gain function G could be theoretically derived as a Bessel function, but the values from the beam chart given by the manufacturer (Fig. 5.1) will be used to keep the characteristics of the model as close as possible to the reality. Then with the values of G applied to Eq. 5.4, it is possible to trace the workspace of the sensor. Considering L = 6m, as given by the manufacturer for ideal reflection, Fig. 5.2 is traced, indicating that the maximal imprecision is about 1.5m (at circa 3.5m range) around the sensor axis. In this way, a workspace is defined in Fig. 5.2, which indicates the area where is possible to have a measure.

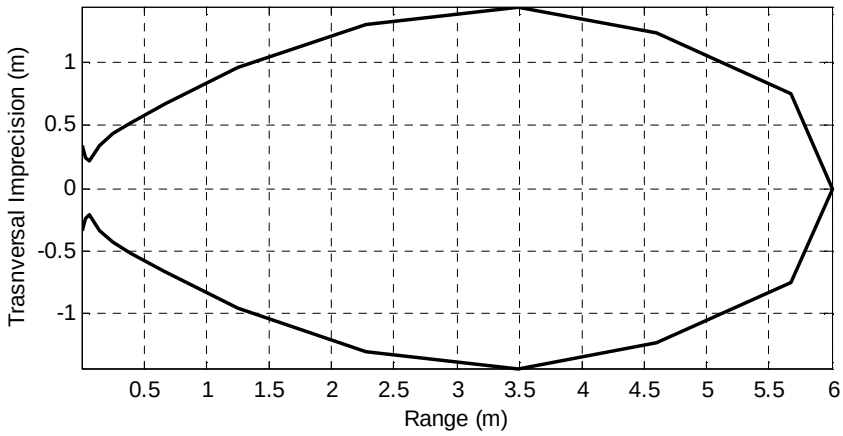

Fig. 5.2: Sensor workspace

Actually the real workspace is in general is much smaller because there are two kinds of reflection: specular and diffuse; in the former the reflected pulse is mirrowed with a reflection angle equal to the incidence angle, in the latter the incidence point becomes instantaneously a new emitter. Usually both of them occur together and, depending on the superface finishing of the obstacle, one in more intensity than the other. When a object is at the lateral of the sensor beam (workspace) it is more probable to have a large incidence angle, then the echo due to specular reflection goes away and just the echo corresponding to the diffuse reflection comes back to the sensor, what is not always enough to be detected.

5.1.2 Fuzzification

A set of three coordinates is necessary to completely localize a point with respect to a frame fixed to the sensor. These numbers are presumed to represent uncertainties of these coordinates too, based on the many sensor characteristics (beam shape, gain, precision…) and

the measured distance. Considering a reference frame fixed to the sensor unit, with its *X*-axis along the sensor emission axis, a set of polar coordinates can be stated as $\{\alpha, \beta, r\}$, where α represents the rotation around the *Z*-axis, followed by a rotation of angle β around the *X*-axis and *r* is the distance to the origin as shown in Fig. 5.3.

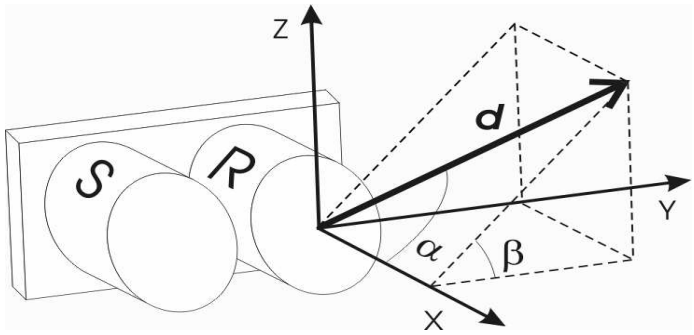

Fig. 5.3 - Sensor Coordinates

The proposition to describe this measurement could be $\varphi_{\alpha,\beta,r}$ = '*at azimuth* α *and X-rotation* β *relative to the sensor's current configuration there is an obstacle present at distance r*'. The certainty of the proposition has to be stated in order to carry on the fusion of the measurements. To obtain such certainty value it would be better first to state the individual certainty of each element of the parameters set $\{\alpha, \beta, r\}$.

If a predicate like '*is a possible value*' is used to define some fuzzy sets, the parameters set could be converted fuzzy sets (or fuzzy numbers) {*A*, *B*, *R*}, where membership degree to each fuzzy set actually would describe '*how possible is that the parameter value is the real value*'. In this way, a set of fuzzy coordinates describing the position of the detected object with its uncertainty is created. To carry on the fuzzification, three of its components have to be established: the type of membership function, the support values and the central values.

There are many different functions possible to be used as membership functions, but considering the distance, as no previous information is available about the error distribution on the parameter *r*, aiming at the simplicity of calculations, the membership function used is a triangular one. The fuzzified *r*, i.e. *R*, has its support directly based on the possible error of the distance measurement; from manufacturer specifications is know that the nominal error is 1% of the measured value, but the error due to temperature change, discussed in last section, will be taken in account: additional 2% are considered for a moderate temperature change (11°C). Then, the fuzzy set *R* is stated as:

- For a given measured distance *d*, the central value is taken as *d* itself.
- The support is considered as ±3% around *d*, represented by a multiplicative factor ε, what means that the actual distance lies in the interval $[d\text{-}\varepsilon.d;\ d\text{+}\varepsilon.d]$, respecting the ranging limits, i.e., the range domain [0; *L*].

- A triangular membership function is used as described in Eq. 5.5 and sketched in Fig.5.4.

$$\mu_R(r) = \begin{cases} 0, & r < (1-\varepsilon)\cdot d \text{ or } r \geq (1+\varepsilon)\cdot d \\ 1-\dfrac{d-r}{\varepsilon \cdot d}, & (1-\varepsilon)\cdot d \leq r < d \\ \dfrac{d-r}{\varepsilon \cdot d}, & d \leq r < (1+\varepsilon)\cdot d \end{cases} \tag{5.5}$$

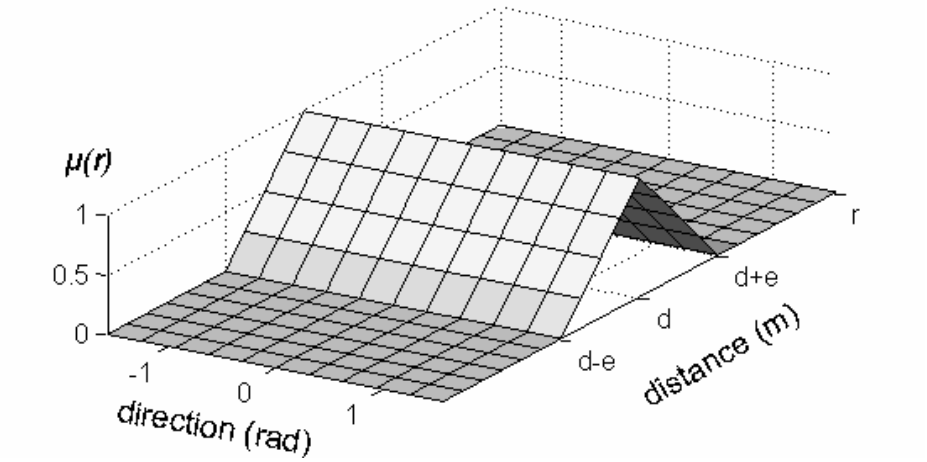

Fig. 5.4: Membership function $\mu_R(r)$ for the distance

To specify a membership function for the parameter α, consider a quotient q between the intensity I_d of an incoming echo and the minimal intensity; from Eqs. 5.3 and 5.4 follows as: $q = I_d / I_m = L^2 \cdot G(\alpha) \cdot d^2$. This quotient represents a safety margin, indicating that it is more possible that a measure with higher intensity is read by the transducer. From this equation, it is clear that for a given measured value d, the quotient and therefore the distribution of possibilities follow the shape of the gain function.

As this gain function originates from a Bessel function, it is acceptable to suppose that it can be approximated by a Gaussian function as described in Eq. 5.6, which is much simpler. Figure 5.5 (left) shows that this supposition is valid; therefore a Gaussian membership function will be used.

$$\hat{G}(\alpha) = \exp\left(-\frac{1}{2}\cdot\left(\frac{\alpha}{\pi/11}\right)^2\right) \tag{5.6}$$

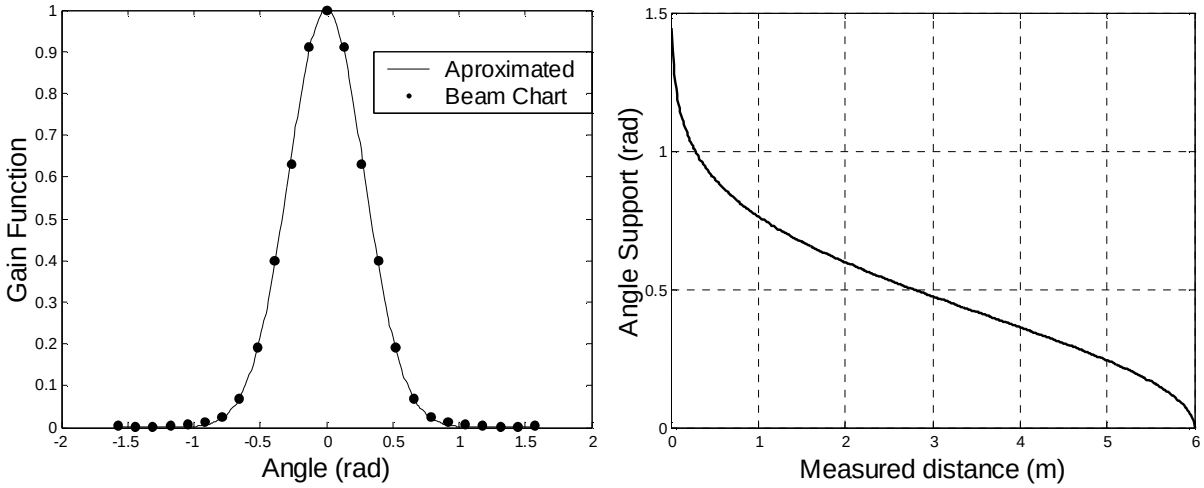

Fig. 5.5: Approximation of *G* by Gaussian function (left); Support for α (right)

Figure 5.2 shows that for a measurement *d*, there is maximal possible value for α, which is on the limit of the workspace at the distance *d*. This limit value can be obtained by the substitution $r_{max} = d$ and $\alpha = \alpha_{max}$ in Eq. 5.4, what leads to $\alpha_{max} = G^{-1}(d^2/L^2)$. Using the approximated gain function from Eq. 5.6, the value of the support as given in Eq. 5.7, results in the curve shown in Fig. 5.5 (right).

$$\alpha_{\max} = \frac{2\pi}{11} \cdot \sqrt{\ln\left(\frac{L}{d}\right)} \tag{5.7}$$

As the support is stated, it is necessary to find the central value: as there is no previous information about the position of the detected object, it is reasonable to consider that it lies on the emission beam, since that would be the most probable direction, given the quotient *q*. So, for the set *A*:

- the central value is taken as 0°;
- the support is given by the interval $[-\alpha_{max}; \alpha_{max}]$, given by Eq. 5.7;
- Gaussian membership function is used as described in Eq. 5.6 and plotted in Fig. 5.6.

$$\mu_A(\alpha) = \exp\left(-\frac{1}{2}\left(\frac{\alpha}{\alpha_{\max}/4}\right)^2\right) = e^{k\cdot\alpha^2} \;\therefore\; k = \frac{242}{\pi^2 \cdot \ln\left(d/L\right)} \tag{5.8}$$

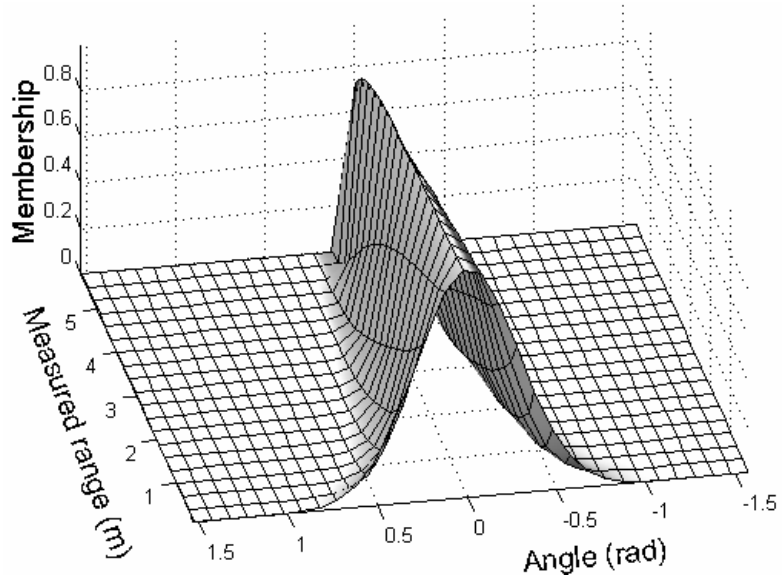

Fig. 5.6: Membership function $\mu_A(\alpha)$

For the angle β the membership function is taken as unitary at its whole domain [0; 2π] because of the simple interpretation of the predicate: any value there is as possible as any other. It is not necessary to state a central value and the support becomes the domain itself because $\mu_B(\beta) = 1$, hence the fuzzy coordinates are stated as folows:

$$\begin{aligned} A &= \{(\alpha, \mu_A(\alpha)) \mid \alpha \in [0; \pi/2]\} \\ B &= \{(\beta, \mu_B(\beta)) \mid \beta \in [0; r_{\max}]\} \\ R &= \{(r, \mu_R(r)) \mid r \in [0; 2\pi]\} \end{aligned} \tag{5.9}$$

Here a fuzzy inverse sensor model was proposed in order to describe the uncertainty degree about a measure. It is similar to the one proposed by Oriolo, Ulivi and Venditelli (1999), based on fuzzy logic too, however the proposition there is related with the occupancy or emptiness of the space, while here it is related with the possibility of a value to be the real one. That leads to slightly different membership functions, aiming at a data fusion process different from the one used here. Moreover, their work was restricted to two-dimensional space, while here a fully three dimensional model was developed.

5.1.3 Common Internal Representation

In the last section, based on a measured distance *d*, the uncertainty about the location of a detected point is stated in the present sensor workspace, for each coordinate. But it is necessary to have a single membership grade for each point **s"** = (α, β, *r*) and not one for each coordinate; so in Eq. 5.10 the Cartesian product is used to find the membership function of the

point **s”**, stating a new fuzzy set *S”*. Since the position of a point is given by the intersection of the isometric lines representing the value of its coordinates, a T-norm is used to combine the fuzzy coordinates through Cartesian Product, resulting in the membership grade of **s”**, as shown in Eq. 5.10. The classical product operation was used as T-norm, in order to make the calculations easier.

$$S'' = A \times B \times R \Leftrightarrow \mu_S = \mu_A \hat{*} \mu_B \hat{*} \mu_R = \mu_A \cdot \mu_B \cdot \mu_R \tag{5.10}$$

Up to now, the natural coordinates of the sensor $\{\alpha, \beta, r\}$ were used. In order to perform a high-level data fusion, it is necessary that this information is available to the robot in a representation common to all sensors; therefore not in their own coordinate systems. It is more convenient to convert them to Cartesian coordinates relative to the reference system fixed on the robot. Observing Fig.5.3 it is easy to obtain the transformation from the natural sensor coordinates to local Cartesian coordinates and then, with the sensor position and orientation relative to the robot, the transformation to a general representation.

From the robot kinematics, the position and orientation of the sensor relative to the robot reference system are always known. The robot will supply the position vector **u** of the origin of the sensor reference frame relative to the robot reference system and the orientation matrix $\mathbf{T} = \begin{bmatrix} \hat{\mathbf{t}}_1 & \hat{\mathbf{t}}_2 & \hat{\mathbf{t}}_3 \end{bmatrix}$ composed of the unitary vectors of the sensor reference frame written in the robot reference system. Hence, the transformation $(\alpha, \beta, r) = \mathbf{h}(x, y, z)$ can be stated for any point (x, y, z) in the robot reference system and conversely $(x, y, z) = \mathbf{h}^{-1}(\alpha, \beta, r)$.

Since α, β and r are associated to membership functions, x, y and z have membership grades too. A way to obtain the corresponding functions is through the application of the *Extension Principle*, discussed in Chapter 3. The direct analysis of this problem can bring some more light into the application of this principle: a point in the sensor workspace has a membership grade, but this membership refers to the set of possible locations of a detected obstacle. Such predicate is invariant with regard to the reference system, what means:

$$\begin{aligned} &(x, y, z) = \mathbf{h}(\alpha, \beta, r) \Rightarrow \mu_S(x, y, z) = \mu_{S''}(\alpha, \beta, r) \\ &\mu_S(x, y, z) = \left(\mu_{S''}\left(\mathbf{h}^{-1}(x, y, z)\right)\right) = \mu_{S''} \circ \mathbf{h}^{-1}(x, y, z) \\ &\qquad = \mu_A \circ \mathbf{h}_\alpha^{-1}(x, y, z) \cdot \mu_R \circ \mathbf{h}_r^{-1}(x, y, z) \end{aligned} \tag{5.11}$$

To calculate the membership values of the right-hand side of Eq. 5.11, it is necessary to know the coordinate transformation $\mathbf{h}^{-1}$, which comes directly from Fig. 5.3. The coordinate r is actually the distance from sensor to the detected point, that is, the norm of **s’** (the detected point position in local Cartesian coordinates). This norm is invariant with respect to the reference system; therefore, it can be calculated using vectors referenced to the robot reference system as in Eq. 5.12. To calculate the angle α, a scalar product is used, which is invariant with respect to the reference system too, leading to Eq. 5.13.

$$r = |\mathbf{s}'| = |\mathbf{s} - \mathbf{u}| = (\mathbf{s} - \mathbf{u})^T \cdot (\mathbf{s} - \mathbf{u}) \tag{5.12}$$

$$\left.\begin{aligned} \hat{\mathbf{e}}_{X'} \cdot \mathbf{s}' &= |\hat{\mathbf{e}}_{X'}| \cdot |\mathbf{s}'| \cdot \cos(\alpha) = |\mathbf{s}'| \cdot \cos(\alpha) \\ \hat{\mathbf{e}}_{X'} \cdot \mathbf{s}' &= \hat{\mathbf{t}}_1^T \cdot (\mathbf{s} - \mathbf{u}) \end{aligned}\right\} \Rightarrow \alpha = \mathrm{a}\cos\left(\frac{\hat{\mathbf{t}}_1^T \cdot (\mathbf{s} - \mathbf{u})}{|\mathbf{s} - \mathbf{u}|}\right) \tag{5.13}$$

By substitution of the membership functions of Eqs. 5.5 and 5.8 in Eq. 5.11, followed by application of the transformations from Eqs. 5.12 and 5.13, the membership function μ_S for points **s** (represented in the robot reference system) is stated. In Fig. 5.7 the resulting membership function is sketched: at the left-hand side, the workspace of the sensor is displayed, where the volume in dark gray represents the volume with non-null membership for a given measure value *d*; at the right-hand side, is the same membership function for constant value of β (a plane.)

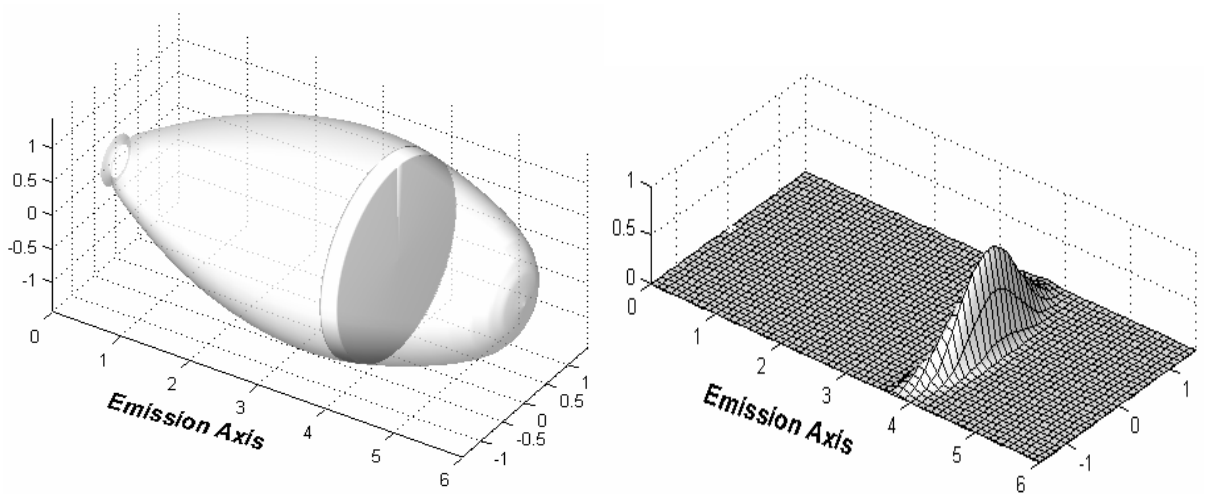

Fig. 5.7: Workspace of sensor (left) and membership value (right)

5.1.4 Topographical Representation

In Eq. 5.11 a three-dimensional membership function with regard to a given measurement is stated. It represents the possibility of each point around the robot to be the detected point. Now this information has to be fused in order to become useful for the robot. Because in this application it is important to describe the topography of the environment, a topographical map in the form $z = f(x, y)$ seems to be most appropriate. Individual measurements will be used to build a map of the surroundings. Since the membership function was stated analytically, there are two possible approaches to perform the fusion:

- Symbolic processing: to execute the fusion calculations symbolically in order to obtain analytically the map representation *f*, that would have as parameters (**T**, **u**, *d*). This approach has the property of retaining all the information, but leads to functions

more and more complicated, hence the symbolic calculations may not always be possible. Moreover, if $f(x, y)$ results too complicated, the symbolic approach may not present a big advantage in terms of processing velocity.

- Numerical processing: it consists of the construction of a three-dimensional grid around the robot, the evaluation of the membership function over this grid and finally the execution of the fusion numerically over this grid. Depending on the grid resolution, it will demand a lot of processing; if the resolution is low in order to accelerate the process, the loss of information may be too high.

Trying to attend the compromise between processing speed and information quality, a hybrid process is proposed here: the 3D membership function will be symbolically projected on the *XY*-plane (where the robot stands), and then this new two-dimensional membership function $\mu_{Sxy}(x, y)$ is evaluated over a grid on the *XY*-plane. As the projection is symbolically obtained, the loss of information at the numerical evaluation is much smaller. Besides, a map is generated with the corresponding membership values. The projection is calculated as shown in Eq 5.14.

$$\begin{aligned}\mu_{Sxy}(x,y) &= \sup_z\left(\mu_S(x,y,z)\right)\\ &= \sup_z\left(\mu_A \circ \mathbf{h}_\alpha^{-1}(x,y,z)\cdot \mu_R \circ \mathbf{h}_r^{-1}(x,y,z)\right)\end{aligned} \tag{5.14}$$

The function $\mu_S(x, y, z)$ is quite complicated; therefore, to obtain symbolically its supremum, the function will be approximated symbolically by a Taylor series of third order around $z = 0$. This value of z is taken because there is no previous information about the terrain where ALDURO will work, what means that the values of z may be completely random (where negative values refers to holes, depressions, etc…).

$$\begin{aligned}&\mu_S(x,y,z) \cong g(z)\\ &g(z) = \mu_S(x,y,0) \;+\; z\cdot\left(\frac{\partial \mu_S(x,y,z)}{\partial z}\right)\Bigg|_{z=0} \;+\\ &\qquad +\; \frac{z^2}{2}\cdot\left(\frac{\partial^2 \mu_S(x,y,z)}{\partial z^2}\right)\Bigg|_{z=0} \;+\; \frac{z^3}{6}\cdot\left(\frac{\partial^3 \mu_S(x,y,z)}{\partial z^3}\right)\Bigg|_{z=0}\\ &g(z) = g_0 + g_1\cdot z + g_2\cdot z^2 + g_3\cdot z^3\end{aligned} \tag{5.15}$$

Equation 5.15 describes the third order polynomial that approximates the membership function μ_S at a generic point (x, y), having the coefficients g_0, g_1, g_2 and g_3 as functions of x and y. As the maxima and minima of a function are determined by finding the roots of its first derivative, it is necessary to solve a second order polynomial equation. As two roots are possible, its necessary to determine which one corresponds to the maximum, what is done by a signal analysis of $g(z)$, as shown in Fig. 5.8, resulting in Eq. 5.16.

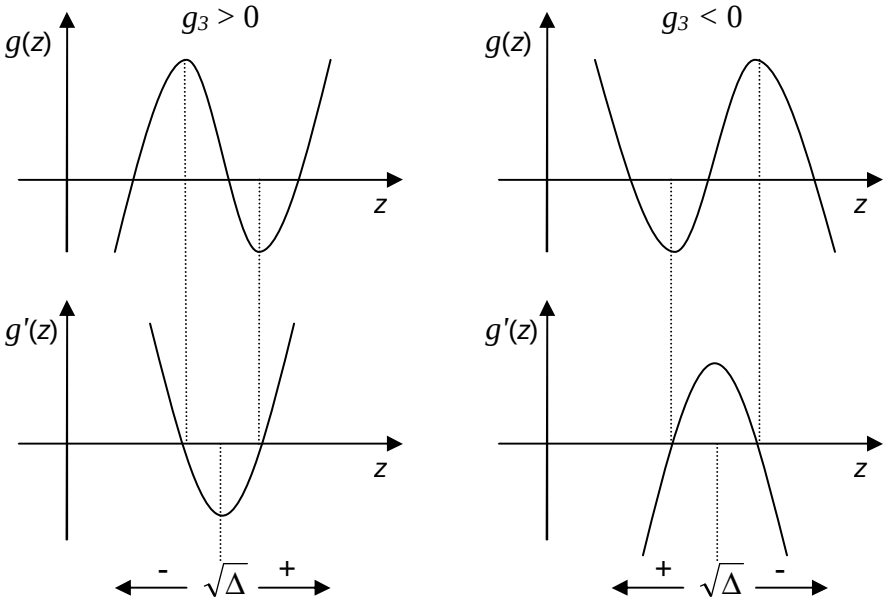

Fig. 5.8: Signal analysis of $g(z)$

$$z_{\max} = \frac{-g_2 - \sqrt{\Delta}}{3 \cdot g_3} \quad \therefore \quad \Delta = g_2^2 - 3 \cdot g_3 \cdot g_1 \tag{5.16}$$

Equation 5.16 is itself the symbolic map $f(x,y)$ originated from one measurement, what is explicitly shown in Eq. 5.17. By substitution of Eq. 5.16 in Eq. 5.14 comes Eq.5.18, which shows the membership function $\mu_{Sxy}(x, y)$ that associates certainty values to the map. In this way, a single measure is converted to a convenient common representation, with a description of the associated certainty, so it is ready to be fused as:

$$z = f(x,y) = \frac{-g_2(x,y) - \sqrt{(g_2(x,y))^2 - 3 \cdot g_3(x,y) \cdot g_1(x,y)}}{3 \cdot g_3(x,y)} \tag{5.17}$$

$$\mu_{Sxy}(x,y) = \mu_A \circ \mathbf{h}_\alpha^{-1}(x,y,z_{\max}) \cdot \mu_R \circ \mathbf{h}_r^{-1}(x,y,z_{\max}) \tag{5.18}$$

5.2 Fusion by TSK (Takagi-Sugeno-Kang System)

Fuzzy inference systems are very often employed to identify models from a data bank or in real time. If the local map of the robot were considered as a system to be identified, inference

systems would be the tool to perform the fusion of the map. As discussed in Chapter 3, a fuzzy inference system is composed by:

- *rules*: they are propositions, which link precedents to specific consequences by means of their firing rate.
- *antecedents*: they are the first part of the rules, composed of one or more fuzzy sets, to which the inputs are compared to, to find their membership grade. Then, operations (*and*, *or*, *complement*) are performed between the sets of same antecedent to obtain the firing rate or (activation value) of the rule.
- *consequences*: they are the second part of the rules. They are combined (union or intersection) according to the firing rate of the corresponding rule, in order to obtain the output of the system.

There are three major groups from inference systems, which are based on the systems: *Mandani*, *Larsen* and *Takagi-Sugeno-Kang* (TSK) [111][115]. The two former groups are characterized by fuzzy inputs and fuzzy outputs, what in general makes necessary the use of a defuzzyfication process (conversion from fuzzy to crisp numbers). The TSK system uses fuzzy inputs too, but has crisp functions as consequences; therefore, it has crisp outputs. Consider the proposition of the rule i, with input x_j, antecedent sets A_{ij} and consequence B_i: 'if x_1 is A_{i1} and x_2 is A_{i2} … and x_n is A_{in} then z_i is B_i'; using this proposition the three systems are below detailed:

- *Mandani*: the used aggregation method is the minimum operator, as shown in Eq. 5.19 for a system with m rules, where w_i is the firing rate of the rule i.

$$\begin{aligned} w_i &= \min\left[\sup_{x_1}\left(\min\left(\mu_X(x_1),\mu_{A_{i1}}(x_1)\right)\right),\ldots,\sup_{x_n}\left(\min\left(\mu_X(x_n),\mu_{A_{in}}(x_n)\right)\right)\right] \\ \mu_Z(z) &= \max_{i=1\ldots m}\left[\min\left(w_i,\mu_{B_i}(z)\right)\right] \end{aligned} \tag{5.19}$$

- *Larsen*: it is an important variation of the Mandani system, where the used aggregation method is the classical product, as described in Eq. 5.20, where z_i^* represents the center of each individual B_i. This system is very often used with a 'center average' as deffuzification method, which becomes simple because of its aggregation method.

$$\begin{aligned} w_i &= \prod_{j=1\ldots n}\left[\sup_{x_j}\left(\mu_X(x_j)\cdot\mu_{A_{ij}}(x_j)\right)\right] \\ \mu_{out}(z) &= \max_{i=1\ldots m}\left(w_i\cdot\mu_{B_i}(z)\right) \\ z_{out} &= \frac{\sum\mu_Y(z)\cdot z}{\sum\mu_Y(z)} = \frac{\sum w_i\cdot z_i^*}{\sum w_i} \end{aligned} \tag{5.20}$$

- *TSK*: the consequence here is not a fuzzy system anymore, but a function, in general a linear function, while the antecedent may be used just as in a Mandani or a Larsen system. So, the proposition corresponding to rule *i* becomes 'if x_1 is A_{i1} … and x_n is A_{in} then $z_i = b_{i0} + b_{i1}.x_1 + \ldots + b_{in}.x_n$', and the output structure is shown in Eq. 5.21.

$$z_i = b_{i0} + b_{i1} \cdot x_1 + \ldots + b_{in} \cdot x_n$$
$$z_{out} = \frac{\sum w_i \cdot z_i}{\sum w_i} \tag{5.21}$$

As the system to be identified is the real local map, the fuzzy sets in the antecedents of the rules have to characterize that; therefore, they will represent the membership grade of a point to a cell in the 2D grid. Besides, the map is the function to be identified; hence, the final output must be the corresponding heights. As most navigation techniques use just a map with crisp values, the output of fusion should be crisp too, in order to have a more general data fusion system, what leads directly to TSK system, whose structure is sketched in Fig. 5.9.

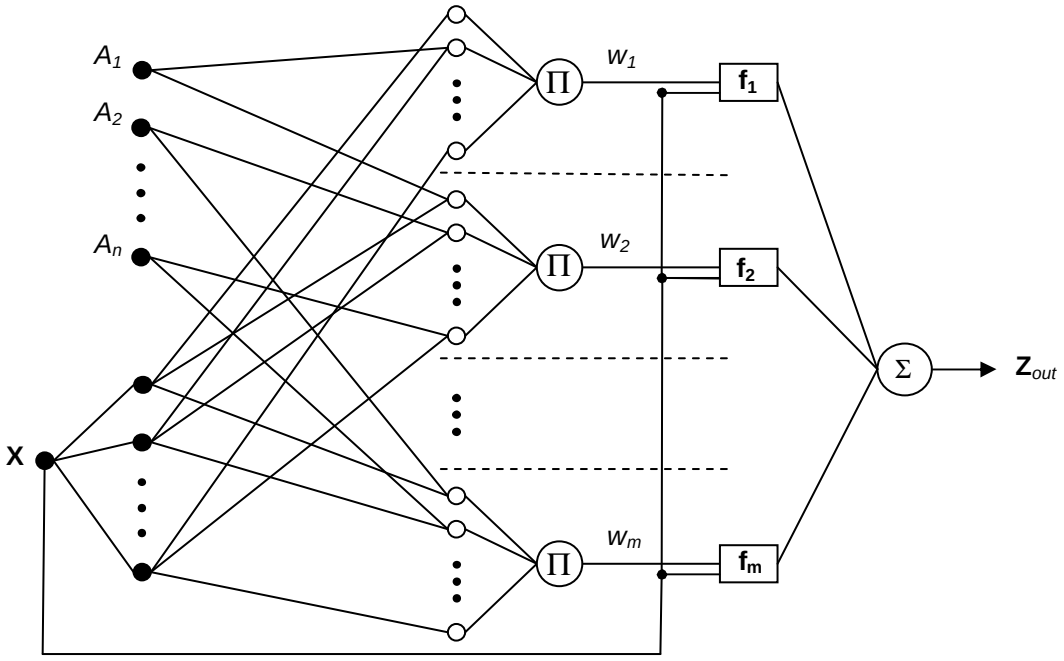

Fig. 5.9: Structure of TSK System

5.2.1 Partitioning

The partition is the group of fuzzy sets defined over the domain of the input variable to characterize this variable. The domain is divided in overlapping sets with membership functions associated to each of them; then, all fuzzy sets so created form the partition of the input domain. A partition is said to be strong if the sum of all membership values of any input

value is 1, otherwise it is called weak. If a partition is strong, it characterizes completely any input of the domain, and no loss of information occurs.

Since the grid uses Cartesian coordinates, a suitable choice for partitioning it is a rectangular partition, in which the axis are partitioned and then Cartesian product is used to obtain the partition of the area. The grid is composed of nodes Q_{ij} that are the center of the rectangular cells C_{ij}, which are overlapping surfaces, whose projections on the axes are overlapping intervals. Fuzzy sets are assigned to each of these intervals and Cartesian product is performed to create bi-dimensional fuzzy sets, each one associated just to one cell, i.e., the partition.

In Eq. 5.22, triangular membership functions (described in Eq. 5.5) are applied on each axis, to make a partition composed of rectangular cells with sides 2λ (in the X direction) and 2σ (in the Y direction), what leads to four different sets of parameters χ for each cell. In Fig. 5.10 the grid was partitioned with $\lambda = 2$ and $\sigma = 2$ resulting in 16 cells.

$$\mu_{C_{ij}}(x,y) = \chi_{k1} \cdot x \cdot y + \chi_{k2} \cdot x + \chi_{k3} \cdot y + \chi_{k4}$$

$$k = \begin{cases} 1, & \left(x_{Q_{ij}} - \lambda\right) \le x < x_{Q_{ij}} \quad and \quad \left(y_{Q_{ij}} - \sigma\right) \le y < y_{Q_{ij}} \\ 2, & \left(x_{Q_{ij}} - \lambda\right) \le x < x_{Q_{ij}} \quad and \quad y_{Q_{ij}} \le y < \left(y_{Q_{ij}} + \sigma\right) \\ 3, & x_{Q_{ij}} \le x < \left(x_{Q_{ij}} + \lambda\right) \quad and \quad \left(y_{Q_{ij}} - \sigma\right) \le y < y_{Q_{ij}} \\ 4, & x_{Q_{ij}} \le x < \left(x_{Q_{ij}} + \lambda\right) \quad and \quad y_{Q_{ij}} \le y < \left(y_{Q_{ij}} + \sigma\right) \end{cases} \tag{5.22}$$

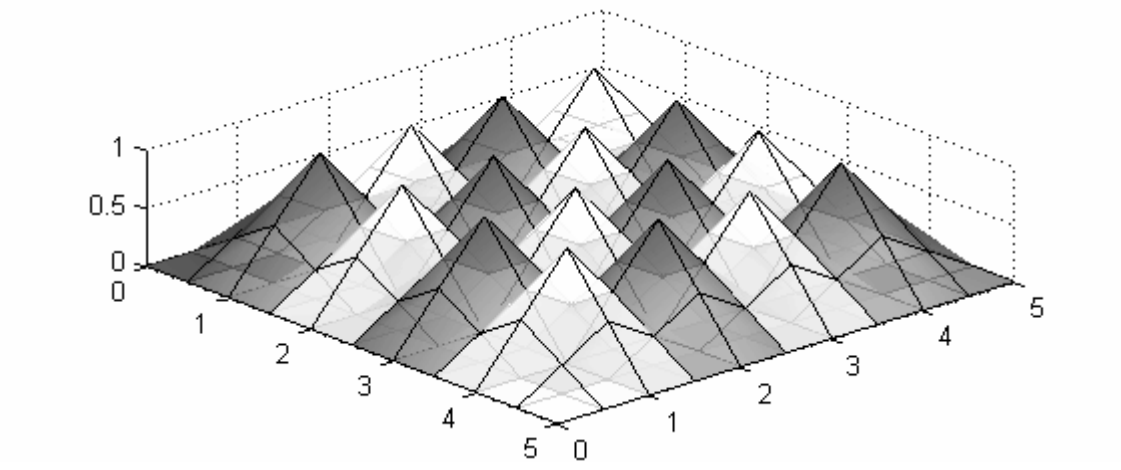

Fig. 5.10: membership functions on 2D grid

If a rule is assigned to each cell, then the firing rate is calculated according to Eq. 5.20, obtaining:

$$w_{ij} = \sup\left(\mu_{S_{xy}}(x,y) \cdot \mu_{C_{ij}}(x,y)\right) \tag{5.23}$$

As $\mu_{Sxy}(x, y)$ is relatively complicated, the calculation of Eq. 5.23 would be pretty difficult, or even impossible, to perform symbolically. Applying the approximation by Taylor series again would lead to more loss of information and the function so obtained would be quite big and complicated too. Considering that $\mu_{Cij}(x, y)$ is continuous by parts, that means that for each cell the whole product in Eq.5.23 would have to be done 4 times, what would need too much processing time.

The further use of symbolic calculations does not seem to be worth anymore, and a change of semantic may state a smooth change to numerical calculations. The propositions at the antecedents of the rules used to state Eq. 5.21 is '(x, y) is C_{ij}', what means 'how much the point (x, y) belongs to the cell ij'; but if the proposition is changed, and the focus is not anymore on the whole area of the grid but on the nodes itself, the new proposition would be '(x, y) is Q_{ij}' and singletons could be used as $\mu_{Cij}(x, y)$.

A singleton is the fuzzy representation of a crisp number. With this, $\mu_{Cij}(x, y)$ would become:

$$\mu_{C_{ij}}(x,y) = \begin{cases} 1, & if \quad (x,y) = Q_{ij} \\ 0, & if \quad (x,y) \neq Q_{ij} \end{cases} \tag{5.24}$$

That would reduce Eq. 5.23 to the simple evaluation of $\mu_{Sxy}(x, y)$ at the considered node. It implies in some loss of information, but guarantees a much faster processing. As the firing rates are available, the output of the system may be calculated as described in Eq. 5.21, where the activation values w_{ij} are used to make a weighted average of the output functions f_{ij}, what is described in Eq. 5.25. This equation gives estimates $\hat{z}$ of the height z, which was already determined in Eq. 5.17 for a single measurement. Now this estimate $\hat{z}$ is calculated by the inference system, which will account for all measurements, producing the global map.

$$\hat{z} = \frac{\sum_i \sum_j w_{ij} \cdot f_{ji}(x,y)}{\sum_i \sum_j w_{ij}} \tag{5.25}$$

The output functions may assume any desired form; however, looking for simplicity, linear functions are used as output: of zero order in Eq. 5.26 and of first order in Eq.5.27. As the used partition is not strong it should be normalized at the summation to obtain the final output, another possibility is to use normalized activation values: $W_{ij} = w_{ij} \Big/ \sum_k \sum_l w_{kl}$. Then, one obtains:

$$\text{zero order}: \quad f_{ij}(x,y) = \theta_{ij}$$
$$\hat{z} = \frac{\sum_i \sum_j \mu_{Sxy}(Q_{ij}) \cdot \theta_{ij}}{\sum_i \sum_j \mu_{Sxy}(Q_{ij})} = \sum_i \sum_j W_{ij} \cdot \theta_{ij} \tag{5.26}$$

$$\text{first order}: \quad f_{ij}(x,y) = \theta_{ij0} + \theta_{ij1} \cdot x_{Q_{ij}} + \theta_{ij2} \cdot y_{Q_{ij}}$$
$$\hat{z} = \frac{\sum_i \sum_j \mu_{Sxy}(Q_{ij}) \cdot \left(\theta_{ij0} + \theta_{ij1} \cdot x_{Q_{ij}} + \theta_{ij2} \cdot y_{Q_{ij}}\right)}{\sum_i \sum_j \mu_{Sxy}(Q_{ij})} \tag{5.27}$$
$$= \sum_i \sum_j W_{ij} \cdot \left(\theta_{ij0} + \theta_{ij1} \cdot x_{Q_{ij}} + \theta_{ij2} \cdot y_{Q_{ij}}\right)$$

5.2.2 Optimization

Equations 5.26 and 5.27 established two possible forms for the inference system, but is still necessary to determine the parameters $\Theta_{ij} = [\theta_{ijk}]$. This an be achieved with methods of optimization by minimizing the error between the estimated height $\hat{z}$ and the height z given by the virtual sensor after each measurement. In this sense, the fusion itself is executed partially by the fuzzy inference system and partially by an optimization process, which here is performed by application of the *Least Squares Estimation* (*LSE*) method. Consider a group of N samples (measurements) to start the optimization and define the following cost function aiming at the minimization of the error as in Eq. 5.28, where $\mathbf{z} = [z_1\ z_2\ \ldots\ z_N]^T$, and the index refers just to the order in which the measurements are taken.

$$J = \frac{1}{N} \cdot \sum_{l=1\ldots N} (z_l - \hat{z}_l)^2 = \frac{1}{N} \cdot [\mathbf{z} - \hat{\mathbf{z}}]^T \cdot [\mathbf{z} - \hat{\mathbf{z}}] \tag{5.28}$$

If a zero-order function is used to find the estimates $\hat{z}$, then Eq. 5.25 becomes Eq. 5.26, which can be rewritten in the matrix form from Eq. 5.29, where P is called the design matrix.

$$\hat{z} = P(x,y) \cdot \Theta \Rightarrow \begin{bmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \vdots \\ \hat{z}_N \end{bmatrix} = \begin{bmatrix} W_{11}^1 & W_{12}^1 & \cdots & W_{mn}^1 \\ W_{11}^2 & W_{12}^2 & & \\ \vdots & & \ddots & \vdots \\ W_{11}^N & W_{12}^N & \cdots & W_{mn}^N \end{bmatrix} \cdot \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \vdots \\ \theta_{mn} \end{bmatrix} \tag{5.29}$$

Substituting it back in Eq. 5.28 and finding the minimum, it is obtained:

$$J = \frac{1}{N} \cdot \left[\mathbf{z} - P(x,y) \cdot \Theta\right]^T \cdot \left[\mathbf{z} - P(x,y) \cdot \Theta\right]$$
$$\nabla_\Theta J = 0 \Rightarrow \Theta = \left(P(x,y)^T \cdot P(x,y)\right)^{-1} \cdot P(x,y)^T \cdot \mathbf{z} \tag{5.30}$$

If first order output functions are used the formula for the optimization is the same as in Eq. 5.30, however the matrices become larger, as described in Eq. 5.31.

$$\begin{bmatrix} \hat{z}_1 \\ \vdots \\ \hat{z}_N \end{bmatrix} = P \cdot \Theta \quad \therefore \quad \begin{matrix} \Theta = \begin{bmatrix} \theta_{110} & \dots & \theta_{mn0} & \theta_{111} & \dots & \theta_{mn1} & \theta_{112} & \dots & \theta_{mn2} \end{bmatrix}^T \\ P = \begin{bmatrix} P_0 & P_1 & P_2 \end{bmatrix} \end{matrix}$$
$$P_0(x,y) = \begin{bmatrix} W_{11}^1 & \cdots & W_{mn}^1 \\ \vdots & \ddots & \vdots \\ W_{11}^N & \cdots & W_{mn}^N \end{bmatrix}$$
$$P_1(x,y) = \begin{bmatrix} W_{11}^1 \cdot x_1 & \cdots & W_{mn}^1 \cdot x_1 \\ \vdots & \ddots & \vdots \\ W_{11}^N \cdot x_N & \cdots & W_{mn}^N \cdot x_N \end{bmatrix} \tag{5.31}$$
$$P_2(x,y) = \begin{bmatrix} W_{11}^1 \cdot y_1 & \cdots & W_{mn}^1 \cdot y_1 \\ \vdots & \ddots & \vdots \\ W_{11}^N \cdot y_N & \cdots & W_{mn}^N \cdot y_N \end{bmatrix}$$

There is no general rule to choose the order of the output function, but in general, a first order function tends to generate smoother surfaces than a zero order does; however, this last one can give a more precise representation of edges and environments with abrupt changes.

By Eq. 5.30, with the appropriated matrices, the optimal parameters can be calculated for a sample of measurements. However, as the data come almost continuously from the sensors, it is interesting to use a recursive form of LSE – Recursive Least Squares Estimator, RLSE – shown in Eq. 5.32, where a is the new line that would be included in the design matrix by the acquisition of one more measurement.

$$P_{k+1} = P_k - \frac{P_k \cdot a_{k+1} \cdot a_{k+1}^T \cdot P_k}{1 + a_{k+1}^T \cdot P_k \cdot a_{k+1}}$$
$$\Theta_{k+1} = \Theta_k + P_{k+1} \cdot a_{k+1} \cdot \left(z_k - a_{k+1}^T \cdot \Theta_k\right)$$
$$a_{k+1} = \begin{cases} \begin{bmatrix} W_{11}^{k+1} & \dots & W_{mn}^{k+1} \end{bmatrix} & \text{zero order} \\ \begin{bmatrix} W_{11}^{k+1} & \dots & W_{11}^{k+1}, & W_{11}^{k+1} \cdot x_{k+1} & \dots & W_{11}^{k+1} \cdot x_{k+1}, \\ & & & W_{11}^{k+1} \cdot y_{k+1} & \dots & W_{11}^{k+1} \cdot y_{k+1} \end{bmatrix} & 1^{\text{st}} \text{ order} \end{cases} \tag{5.32}$$

To start, P is set to $10^7.I_{mn\times mn}$ and Θ to 0. In next section, it will be explained how the map is moved together with the robot and how the new values for Θ are calculated, while the matrix P is just reset. That represents no problem, since the RLSE has a very fast convergence.

In Fig. 5.11, it is possible to see the comparison between a simulated ground and as it would be perceived by the robot. The approximation can be improved by a higher resolution of the grid, what, however, leads to an augmentation of the processing time. A compromise between processing demand and map quality has to be respected when choosing the number of partitions and the order of output functions to be used.

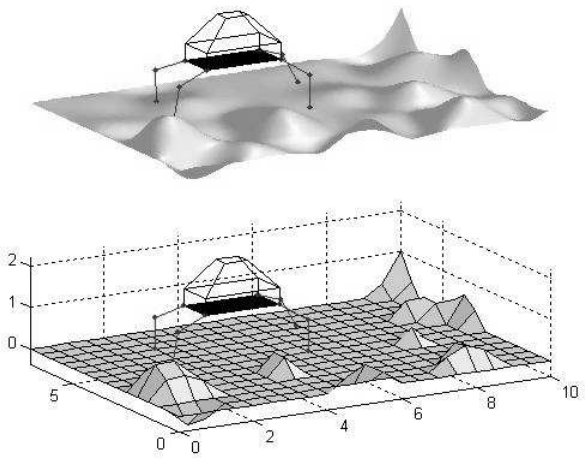

Fig. 5.11 - Robot on simulated ground (over) and perceived ground (below)

5.2.3 Map Motion

In the last section, a map of the robot's surroundings was constructed by use of a grid under the robot. The map is just a local representation of the environment; therefore, as the ALDURO walks, it is necessary to keep the map up to date, with information about the new place towards which the machine goes or as a local observer on ALDURO would view information about the place (and respective obstacles) that comes towards the robot. Then, the map is considered as moving under the robot and the necessary information is extracted through the grid, which is linked to the robot.

To construct the grid, the starting point is the *platform fixed* reference system K_P, which is fixed to the center of the robot body. With information supplied by the *Kinematics Module* of ALDURO, acquired by an on-board inclinometer, it is possible to state the rotation matrix ${}_Q\mathbf{T}_P$, which is the transformation from K_P into K_Q, the *Quasi-Global* reference system, whose *XY*-plane is parallel to the horizontal plane and the origin is coincident with the one from K_P. The *Kinematics Module* supplies as well a nominal height h_R of the robot, calculated from the

prescribed position of the standing feet relative to the body; then, the frame from K_P is translated down in the vertical direction to obtain the *Grid* reference frame K_G. In this way, by Eq. 5.33 a vector $\mathbf{u}_P$ (e.g. the position of a foot relative to the platform reference system) can be represented in the Grid referential as $\mathbf{u}_G$.

$$\mathbf{u}_G = {}_Q\mathbf{T}_P \cdot \mathbf{u}_P + \begin{bmatrix} 0 & 0 & -h_R \end{bmatrix}^T \tag{5.33}$$

A grid composed of rectangular cells, with $n \times m$ nodes Q_{ij} ($i = 1, \ldots, n$; $j = 1, \ldots, m$) is created on the *XY*-plane of the *Grid* reference frame, with its center coincident with the origin of K_G. It is interesting to note that, in despite of changes in robot posture, this grid remains parallel to the horizontal plane but its distance to the robot body is variable, depending on the configuration of the legs. As the robot moves, the grid goes along with it and as the robot changes its direction, the grid changes its direction too.

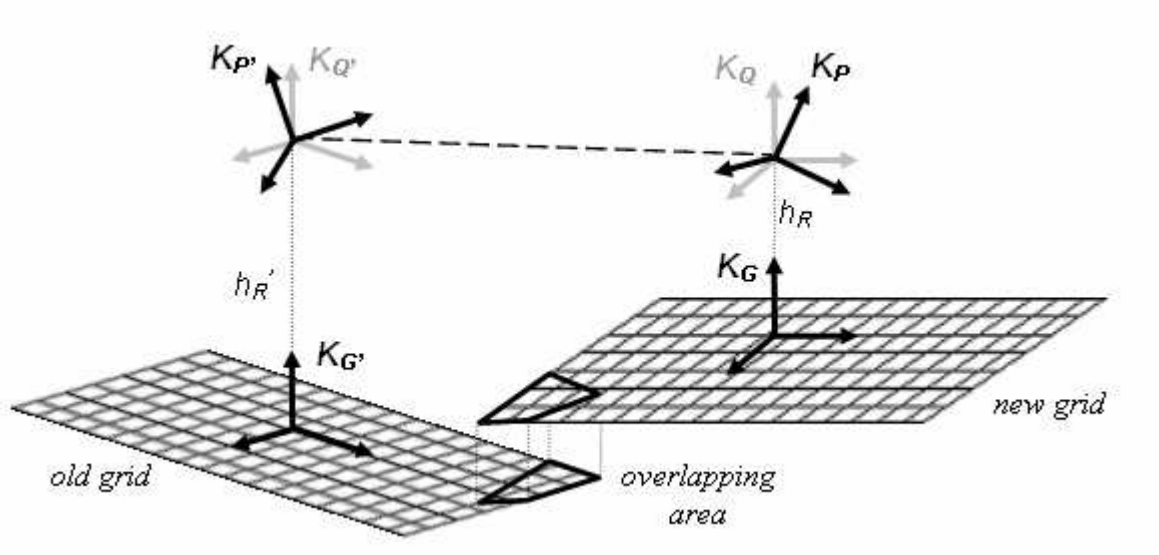

Fig. 5.12: Grid Motion

If time is sampled at discrete points; then, for each given time point, a map composed of points M_{ij} is stated by estimating heights $\hat{z}_{ij}$ to the nodes Q_{ij}, that is, $M_{ij} = [Q_{xij}, Q_{yij}, \hat{z}_{ij}]^T$. In the next time point the robot was already submitted to a small rotation ${}_P\mathbf{T}_{P'}$ and a small translation $\boldsymbol{\tau}_P$, represented in the new body fixed frame. It must be assumed that rotation and translation are small enough such as to guarantee that the general robot configuration is not changed between the two considered time points, because three points with fixed position in a inertial referential are necessary to calculate ${}_P\mathbf{T}_{P'}$ and $\boldsymbol{\tau}_P$. By taking the standing feet as the fixed points, with their new positions $\mathbf{u}_k$ and old positions $\mathbf{u}_k'$, the rotation and translation of robot body can be calculated by the 9×9 linear system in Eq. 5.34, where the sign (') denotes the values in the past moment, like in Fig. 5.12. In order to make this calculation always

possible, at least three of the standing feet must be the same in the two considered time point, what is actually no problem, since ALDURO stands on the four feet before to change the swing foot; enabling the change of the considered set of legs. Thus one obtains:

$$\begin{cases} \mathbf{u}_1 = {}_P\mathbf{T}_{P'} \cdot \mathbf{u}'_1 + \boldsymbol{\tau}_P \\ \mathbf{u}_2 = {}_P\mathbf{T}_{P'} \cdot \mathbf{u}'_2 + \boldsymbol{\tau}_P \\ \mathbf{u}_3 = {}_P\mathbf{T}_{P'} \cdot \mathbf{u}'_3 + \boldsymbol{\tau}_P \end{cases} \tag{5.34}$$

As the grid does not undergo the same rotation as K_P (it must remain parallel to the horizontal plane), it is easier just to construct a new grid, instead of transforming it. Besides, information of the old map $M'_{G'}$ has to be represented in the new map M_G; to do that, the new grid Q_G is sequentially transformed from K_G to K_Q, K_P, $K_{P'}$, $K_{Q'}$ and finally to $K_{G'}$ to obtain $Q_{G'}$ as shown in Eq. 5.35, where an eventual change in the nominal height of the robot is considered. The points of the new grid represented in $K_{G'}$ are used to generate the new map $M_{G'}$ in the old grid referential, what is done through the TSK system; then this new map must be represented in K_G, follows:

$$Q'_{G'} = {}_{Q'}\mathbf{T}_{P'} \cdot {}_P\mathbf{T}^T_{P'} \cdot \left({}_Q\mathbf{T}^T_P \cdot \begin{bmatrix} Q_{Gx} & Q_{Gy} & 0 \end{bmatrix}^T - \boldsymbol{\tau}_P \right) + \begin{bmatrix} 0 & 0 & h_R - h'_R \end{bmatrix}^T \tag{5.35}$$

$$M_G = {}_Q\mathbf{T}_P \cdot \left({}_P\mathbf{T}_{P'} \cdot {}_{Q'}\mathbf{T}^T_{P'} \cdot M_{G'} + \boldsymbol{\tau}_P \right) + \begin{bmatrix} 0 & 0 & h_R - h'_R \end{bmatrix}^T \tag{5.36}$$

With the the data of M_G and new incoming measurements, fused by the TSK system and the Recursive LSE, the map is updated to the new surroundings, following the flow from the diagram in Fig. 5.13.

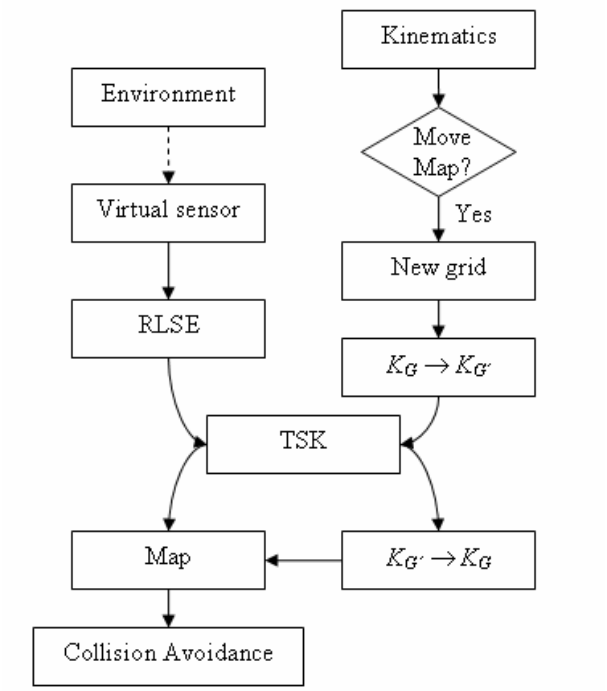

Fig. 5.13: Flow diagram of Map Building

5.3 Navigation

Navigation techniques can be classified within two major groups: reactive and deliberative. The proposed solution for ALDURO is a mix of both [34]. Actually, the incremental building of a dynamic map of the environment (a typical deliberative task) is employed together to the formulation of local paths (typically reactive).

There are many different techniques to carry out reactive motion planning, and for this particular application, some peculiarities are important when choosing it:

- The legs of the robot make full spatial movements.
- Each leg has to fulfill the kinematical constraints imposed by the body movement.
- The reference values for the motion control system of ALDURO are velocities. Therefore, the collision avoidance module has to generate corrective velocities as outputs, to supply the motion generator with the information about the necessary deviation.

The controller developed here is based on the concept of *General Perception* [15][17][18], which is usually employed at typically reactive systems, just like 'wall followers' and 'line followers', to characterize the situation in which the robot finds itself. In the original conception of general perception, the robot is guided by a representation of its perception only, no map of the environment is used and walls and obstacles are not modeled either. Here, instead of taking directly the row data supplied by the perception, the map generated by the data fusion is used in order to provide complete information for this reasoning phase.

The map is composed of points M_{ij}, in the reference frame K_G as explained in the last section, where the *XY*-plane should coincide with the ground if it is flat. The map points are classified in two groups, according to their height: the *high* ones and the *low* ones. The group *low* comprehends all points with height lying between the upper limit h_{max} and the lower limit h_{min}, while the group *high* comprehends the points outside these limits:

$$\left.\begin{aligned} M_{ij} \in low &\leftrightarrow h_{\min} \le \hat{z}_{ij} \le h_{\max} \\ M_{ij} \in high &\leftrightarrow \left(\hat{z}_{ij} < h_{\min}\right) \cup \left(\hat{z}_{ij} > h_{\max}\right) \end{aligned}\right\} \rightarrow low = \neg high \tag{5.37}$$

The classification of the terrain is done considering the constructive characteristics of ALDURO, as shown in Fig. 5.14. Points of set *low* are those for which it is possible for the robot to negotiate them or even stand on them if necessary; but, as the robot does not walk on a flat surface, they have to be considered in order to perform a collision-free walking. The set

high represents those points that are too high to come over. To state these sets some measures are important, where three are different vertical distances measured from robot body to foot:

- h_{leg}: the leg is completely extended downwards, it is the maximal distance from the platform that the foot can reach;
- h_{sec}: the leg is bent, it is the highest possible position for the foot;
- h_R: nominal distance from body to ground (already explained in last section).

Based on such measures, three other heights, relative to the grid, are calculated:

- $h_{max} = h_R - h_{sec}$: represents the highest obstacle possible to overcome;
- $h_{min} = h_{leg} - h_R$: represents the deepest depression possible to ALDURO to step into;
- $h_T = h_{max} - h_{min}$: is the vertical workspace of the foot.

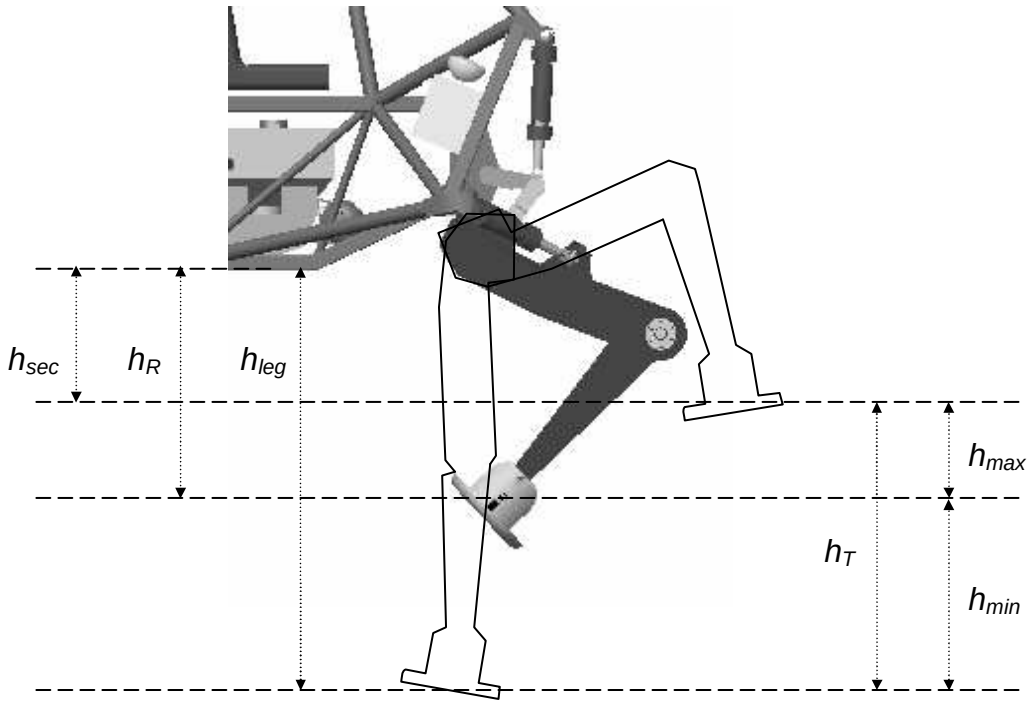

Fig. 5.14: Terrain classification

As the shanks are the most exposed part of the robot, that is, the most probable part to undergo a collision, the collision avoidance should consider the whole shank for each leg. However, the *Motion Generation* module of ALDURO just receives the feet positions as inputs (like there was no knee and hip joints), therefore, only feet can be considered when avoiding collisions. In most cases there is no problem, because the map is a convex surface,

then, if the foot is 'safe', the whole shank is safe from collisions too. Given the constructive characteristics from ALDURO, the foot is always the lowest point of the leg, what means that, if this point is higher than the surrounding points of the convex surface, the whole shank is higher too.

The reference point O_k (k = 1,…,4) is stated at each of the feet and for every point M_{ij} there is assigned a perception ρ_{ij}, which is composed of a vertical component and a horizontal one, yielding $\rho_{ij} = \rho_{ij}^V \cdot \rho_{ij}^H$. The horizontal component is actually the quadratic normalized horizontal distance between the points on the map and the reference point O_k. Normalization is done by dividing by the maximal distance D on the grid (half of its diagonal):

$$\rho_{ijk}^H = \begin{cases} 0, & if \ \left|M_{ij} - O_k\right| \geq D \\ 1 - \dfrac{\left(M_{ij} - O_k\right)^T \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \left(M_{ij} - O_k\right)}{D^2}, & otherwise \end{cases} \tag{5.39}$$

The vertical component is calculated in a similar manner: it is the normalized vertical oriented distance between a point in the foot workspace and the reference O_k. This distance may vary form $-h_T$ to $+h_T$, and its square can be used in order to enhance the influence of points a little farther, giving some prediction ability to the controller. Thus it holds:

$$\rho_{ijk}^V = \begin{cases} 1, & if \ M_{ij} \in high \\ \dfrac{[0 \quad 0 \quad 1] \cdot \left(M_{ij} - O_k\right) + h_T}{2 \cdot h_T}, & otherwise \end{cases} \tag{5.40}$$

$$\boldsymbol{\rho}_{ijk} = -\rho_{ijk} \cdot \mathbf{B} \cdot \frac{\left(M_{ij} - O_k\right)}{\left|M_{ij} - O_k\right|};$$

$$\mathbf{B} = \begin{cases} \sqrt{2} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, & if \ M_{ij} \in high \\ \sqrt{3} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, & if \ M_{ij} \in low \ and \ M_{Zij} > O_{Zij} \\ \sqrt{3} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, & if \ M_{ij} \in low \ and \ M_{Zij} \leq O_{Zij} \end{cases} \tag{5.41}$$

The perceptions are then used to calculate the perception vector assigned to each point in the map. A normal selection matrix **B** is used to make that point of group *high* have just influence

in the horizontal motion, while points of group low have a vertical influence too, always pushing the feet out of the obstacles, what makes two different forms of **B** for points above and below the foot necessary, as shown in Eq. 5.41.

For a given reference point *O*, the general perception vector **ρ** is composed of all individual perceptions $\boldsymbol{\rho}_{ij}$:

$$\boldsymbol{\rho}_k = \max_{i,j}\left(\rho_{ijk}\right) \cdot \frac{\sum_i \sum_j \boldsymbol{\rho}_{ijk}}{\sum_i \sum_j \rho_{ijk}} \tag{5.42}$$

Its direction equals the sum of the perceptions relative to each point in the map and its length equals the strongest individual perception. Since the robot is moving, change in time of the general perception should be considered. For simplicity, just the change in its modulus is considered, which can be easily obtained by derivation of the individual perception. It holds:

$$\dot{\rho}_{ijk} = \dot{\rho}^V_{ijk} \cdot \rho^H_{ijk} + \rho^V_{ijk} \cdot \dot{\rho}^H_{ijk} \tag{5.43}$$

$$\dot{\rho}^H_{ijk} = \begin{cases} 0, & if \ \left|M_{ij} - O_k\right| > D \\ \dfrac{2}{D^2} \cdot \left(M_{ij} - O_k\right)^T \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \dot{O}_k, & otherwise \end{cases} \tag{5.44}$$

$$\dot{\rho}^V_{ijk} = \begin{cases} 0, & if \ M_{ij} \in high \\ -\dfrac{\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot \dot{O}_k}{2 \cdot h_T}, & otherwise \end{cases} \tag{5.45}$$

The perception's change in time is related to the maximal velocity of the shank. Given the walking pattern of ALDURO, i.e., one swing leg and three standing ones at each step, the average relative velocity between legs and body is equal null. Therefore, the maximal leg absolute velocity V_{Lmax} is four times the maximal robot absolute velocity V_{Rmax}, and the maximal perception change rate is given by:

$$\dot{\rho}_{\max} = V^2_{L\max} / D^2 = 16 \cdot V^2_{R\max} / D^2 \tag{5.46}$$

Next, a normalized perception change rate $\dot{\rho}^*_{ijk}$ is calculated as:

$$\dot{\rho}^*_{ijk} = \begin{cases} \dfrac{\dot{\rho}_{ijk}}{\dot{\rho}_{\max}} = \dfrac{D^2 \cdot \dot{\rho}_{ijk}}{16 \cdot V^2_{R\max}}; & if \ \dot{\rho}_{ijk} \geq 0 \\ 0; & if \ \dot{\rho}_{ijk} < 0 \end{cases} \tag{5.47}$$

Then the *maximum* operator is applied as in Eq. 5.48 to find the perception change rate of each leg.

$$\dot{\boldsymbol{\rho}}_k^* = \max_{i,j}\left(\dot{\rho}_{ijk}^*\right) \tag{5.48}$$

As just one foot moves at once and because the *Motion Generator* considers just the movements of the feet [46], it is clear that it would not be worth to employ the collision avoidance technique at other than the swinging foot. The index *k*, referring to the legs, was used in the above equations just to show that the technique is more general, being able to be applied at a system where the movement of other parts than the feet is considered, calculating simultaneously the perception of many moveable parts.

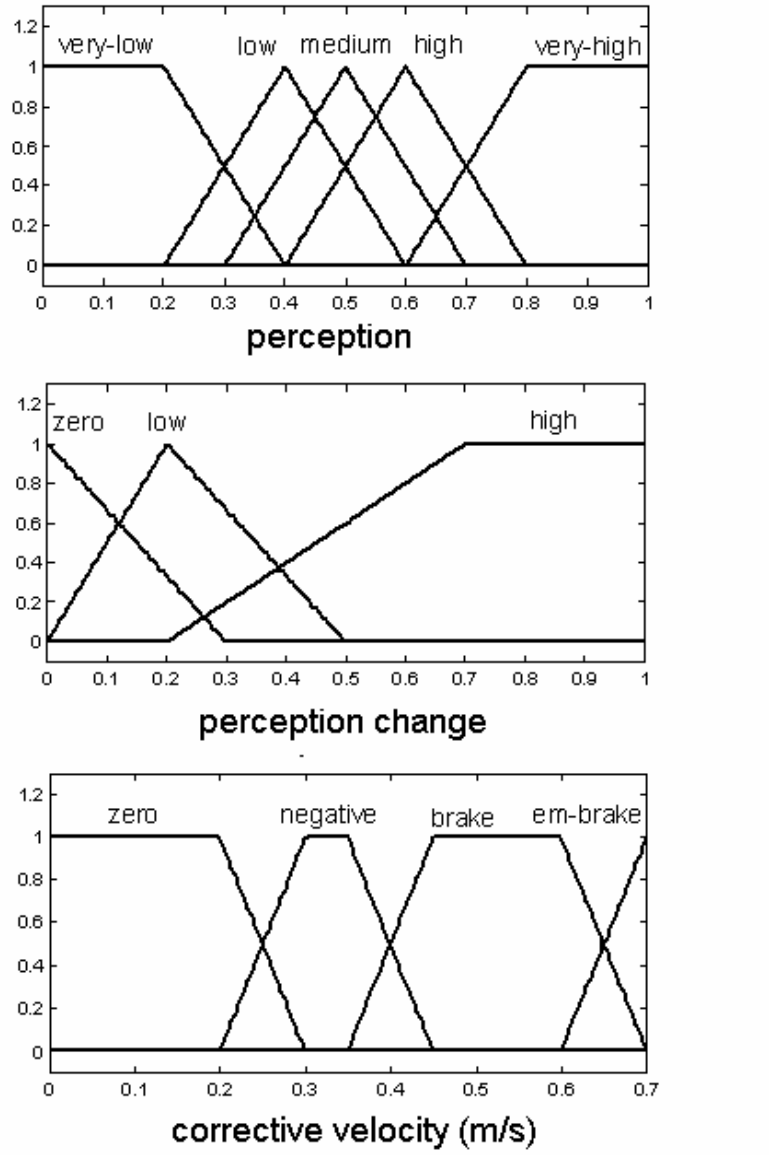

Fig. 5.15: Membership functions of the controller

A fuzzy controller [16] is used to combine both the general perception vector and the change rate of its modulus in order to generate a corrective velocity each foot. The direction used for the corrective velocity is the one of the perception vector; in the controller, just its length is considered. The input sets are shown in Fig. 5.15, and the control surface in Fig 5.16, show the output, i.e., the modulus of the corrective velocity.

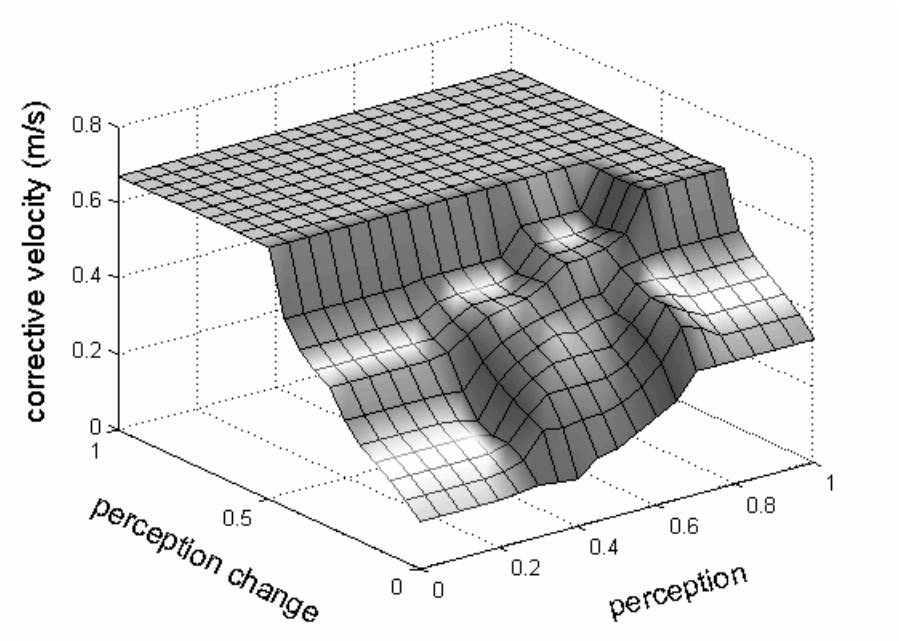

Fig. 5.16: Control surface

The corrective velocity is really just a correction at the reference velocity for the foot. As the motion of the leg is calculated by the *Motion Generator*, some influence may be transmitted to the robot body, because each leg has four degrees of freedom corresponding each one to an actuator, each one with well-known limits. When an actuator is close to its limit, the *Motion Generator* will correct the robot body movement in such a way to make possible to the leg to continue its own movement. That works like a reaction of the body to the movement of the legs at their limit.

6 Realization

To implement the proposed collision avoidance system, a suitable ultrasonic range finder and all the hardware necessary to support it have to be specified. This specification is oriented to the application on ALDURO; therefore, some constraints relative to the project must be observed, like large area to cover (due to the robot dimensions) and the kind of environment where the robot is supposed to work. Besides, the placement of the sensors over the robot has to be analyzed too, in order to provide the maximal covering of the surroundings and to avoid false measures of the sensors caused by the many movable parts of the robot if the placement is not carefully stated.

6.1 Selection of Ultrasonic Range Finder

Many models of ultrasonic sensors, from different manufactures, are available. The main points to be analyzed were: beam pattern, range, cost and interface to the robot's hardware. The beam pattern together with the range, determine the workspace of the sensor, i.e., the volume where a successful detection may occur. In Fig. 5.2 is possible to see that the ultrasonic beam actually presents a conical form near to the sensor, and finishes at a quasi-parabolic curve. The conical form is due to the directivity in overall sensitivity of the sensor, the so-called beam pattern, and can be slightly modified by adaptations in the geometrical characteristics of the sensor package.

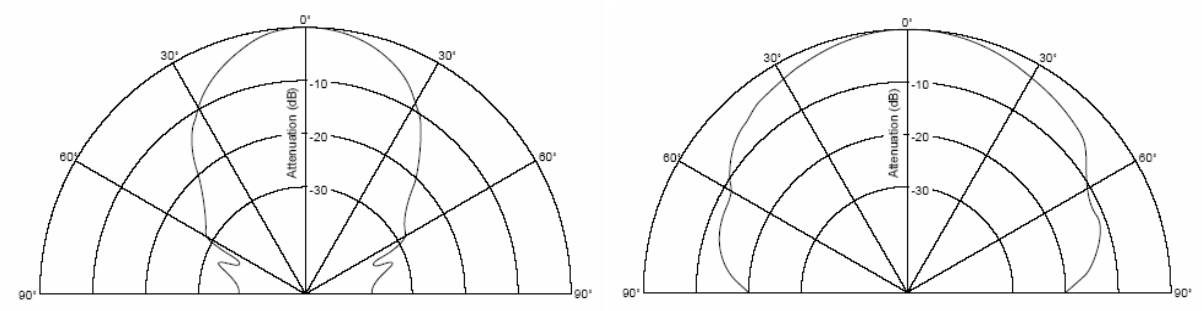

Fig. 6.1: Directivity in Sound Pressure Level at 40 kHz for senders MA40B8S (left) and MA40S4S (right), from Murata

As the measurement process consists of emission and reception of ultrasonic pulses, the different properties of the transducers, relative to each of these tasks, have to be considered.

When the pulse is emitted, it can be considered as a spherical wave with variable intensity along the wave front, because of a directional attenuation as shown in Fig. 6.1. Therefore, depending on the direction relative to the sensor, objects at the same distance of the sender will receive a different amount of energy, so they will generate echoes of different intensities. The sound pressure level corresponds to the intensity of the ultrasonic pulse.

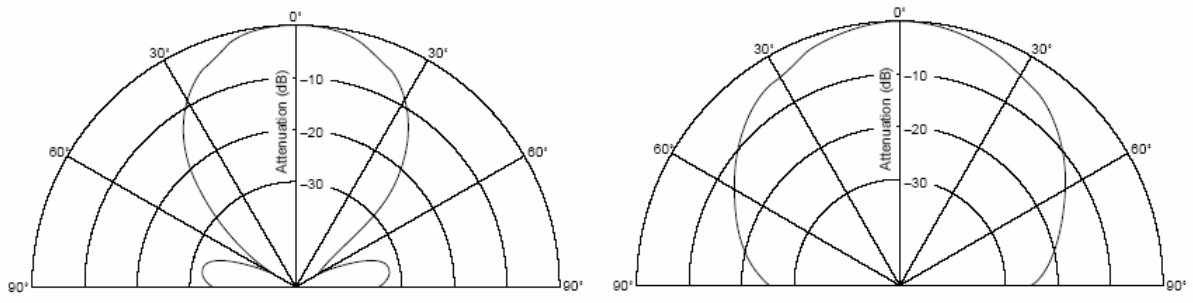

Fig. 6.2: Directivity in Sensitivity at 40k Hz for receivers MA40B8R (left) and MA40S4R (right), both from Murata

In the same way, receivers present a directional sensitivity too; incoming signals are attenuated at a gain that is dependent of the incoming direction. It means that incoming echoes, with same intensity but different directions, will suffer different attenuations, as shown in Fig. 6.2. If the distance between sender and receiver is small compared to the ranges to be measured, it can be neglected and the sensor is considered as a single transducer, then just one chart is considered: the directivity in overall sensitivity. As in a real measurement, the direction towards the reflecting point and the direction of the echo's return are the same, the curve for the directivity overall sensitivity can be obtained directly from the product of the directivity curves for sender and receiver.

The beam pattern for three different sensors are shown in Fig. 6.3, where it is possible to see that beams may vary from narrow (Fig. 6.3 left) to wide (Fig.6.3 right). In general, it is interesting to have a beam as narrow as possible, what makes it easier to determine the direction of the detected object, but in this application, because of the robot dimensions, it is necessary to cover a volume as big as possible with a single measurement, in order to minimize the number of necessary sensors. That would lead to a sensor with wide beam, but that is not completely desirable, because the information would be too diffuse, what could bring some quality degradation to the map; therefore a sensor with middle beam width seems appropriate, what means a sensor with a cone angle of about 20°.

The ultrasound frequency used is important too. With higher frequencies is possible to obtain a narrower beam but it tends to cause specular reflections, so a sensor that uses ultrasound at the 235 kHz looking at a hard flat surface at an angle of more than 8° or 9° do not yield an

echo from that surface, what can cause multiple reflections too. Soft surfaces do not reflect well at 235 kHz and they might not be detected, but hard round objects are well detected. As the 40 kHz sensors have a good detection against soft materials (including vegetation) and a relative longer range, they are appropriated when the environment may present any kind of objects. In Fig. 6.3, beam patterns of three different sensors are shown, for SRF235, SRF08 and SRF10 (from left to right). The former works with ultrasound at a frequency of 235 kHz, while the two later use 40 kHz of frequency.

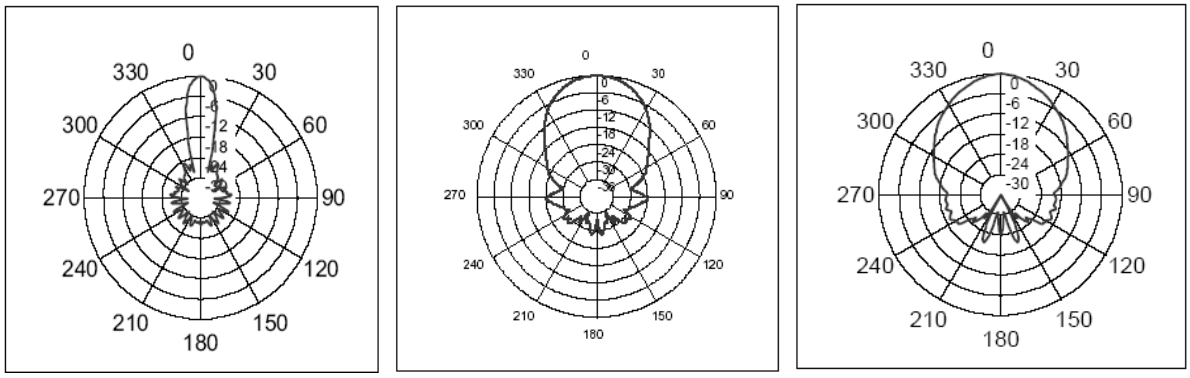

Fig. 6.3 – Directivity in Overall Sensitivity of SRF235, SRF08 and SRF10, from Devantech

The quasi-parabolic form of the beam at its end is due to the quadratic decrease of the sound intensity, combined to the beam chart. That defines the maximal range, what is very important when specifying any sensor, because this range separates the interesting space from the not interesting space. As the local map is the base for the collision avoidance system, it must comprehend the whole interesting area but, in order to define the dimensions of this area, it becomes important to state how long a foot can travel at each step.

The relative maximal step length of ALDURO is about 1,45m, but because of the robot movement, the foot may travel a longer distance at each step. As just one leg moves at each time, the absolute average velocity of the legs is four times greater than the absolute average velocity of the robot body and the relative velocity between body and moving leg is three times the body velocity. Moreover, the absolute distance crossed by the leg is 4/3 of the relative one, i.e., 1,93m per step. The map corresponding to the space relative to two steps forward has to be available when a step starts; so that collisions of the leg, when performing a step, can be avoided and a suitable position to land the foot (aiming at the easy of the next step) can be chosen. That means that at least 3,86m forward on the environment have to be know, and the suitable minimal range (with a safety factor) for the ultrasonic sensors becomes 4m.

The maximal expected velocity for ALDURO is 5km/h (1,39m/s), which is relatively low. Considering a stable gait at the maximal velocity and maximal step length, the feet would travel at a maximal average velocity of 5,56m/s, what means that a step at this extreme condition would take 0,35s and the time between two consecutive steps of the same leg is 1,04s. To cover 4m twice (sent pulse and returning echo), an ultrasonic pulse takes about 24ms; that means that between two consecutive steps it would be possible to make 44 measurements with one sensor.

There are many different ultrasonic sensor systems available for distance measurement, where the signal is pre-processed to filter noise and cross talking. But the devices used for such task make the system much more expensive. Moreover, it is important to test how robust the map building system proposed here is, when using signals without any preprocessing. Another important topic directly related to the cost is the package: there are different kinds of package, with different security levels. For the prototype, it is interesting to have complete access to the sensor. Moreover, since the proposed data fusion process just uses the distance measured values and not the echo shape, it would be interesting to have a unit with digital output.

In sumary, the desired technical characteristics for the range finder unit would be:

- Beam angle about 20°
- Maximal Range of at least 4m
- Digital output

Considering that, the ultrasonic sensor SRF08 of Devantech (Fig.6.4) was chosen. Its maximal range of 6 m is appropriate for this application as well as its middle radiation beam, with cone angle about 45° (half cone angle of 22°30'). Besides, the communication is done via the I^2C-bus, available on a wide variety of microcontrollers. Therefore, a microcontroller can be used as interface with the main controller, stating a low-cost and simple solution.

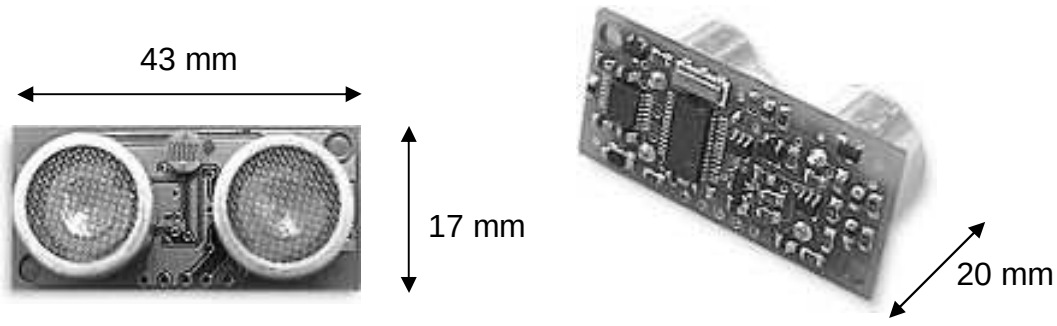

Fig. 6.4: Dimensions of the SFR08

6.2 Interface

As mentioned, the SRF08 uses the Inter Integrated Circuit (I^2C) Bus to communicate. This control bus is designed to provide communication between integrated circuits in a system. As the main controller of ALDURO is a PC, one possibility to make the interface would be to use an I/O board which provides an I^2C port for the computer; another option would be to use a microcontroller that communicates through interfaces I^2C-bus and a USB, serial or parallel port.

With this microcontroller, all the management of the sensors, as firing, waiting for answer and sensor setup, can be done completely independent of the main controller, which just receives the measured values.

6.2.1 The Bus

Developed by Philips in the early 1980s, I^2C is a simple two-wire bus with a software-defined protocol and consists of two simple lines connecting all the ICs in a system. Any I²C device can be attached to a common I²C-bus and even though the system changes, such devices can be easily added or removed without affecting other parts of the system. That gives a very modular structure to the collision avoidance system, enabling the addition or removal of sensors without difficult.

Its software-controlled addressing scheme eliminates the need for address-decoding hardware. In this scheme, each device has a unique 7-bit I²C address, so that it is always known specifically who is communicating. A 7-bit addressing allows up to 128 devices on the same bus, but some of these addresses are reserved for special commands so the practical limit is around 120. The used terminology in an I^2C network comprehends:

- *Transmitter*: the device that sends data to the bus.
- *Receiver*: the device that receives data from the bus.
- *Master*: the component that initializes a transfer; generates the clock signal (SCL) and terminates the transfer. A master can be either a transmitter or a receiver.
- *Slave*: the device addressed by the master. A slave can be either receiver or transmitter.
- *SDA*: data signal line (Serial Data)
- *SCL*: clock signal line (Serial Clock)

Any I²C device can be attached to an I²C-bus and every device can talk with any master, passing information back and forth. There needs to be at least one master, which generates the clock information of the network to make its synchronization and sends it through the SCL line. The data is sent by the SDA line and, in spite of being a two wire bus, in general four wires are used, to supply energy too (power and ground), as in Fig. 6.5.

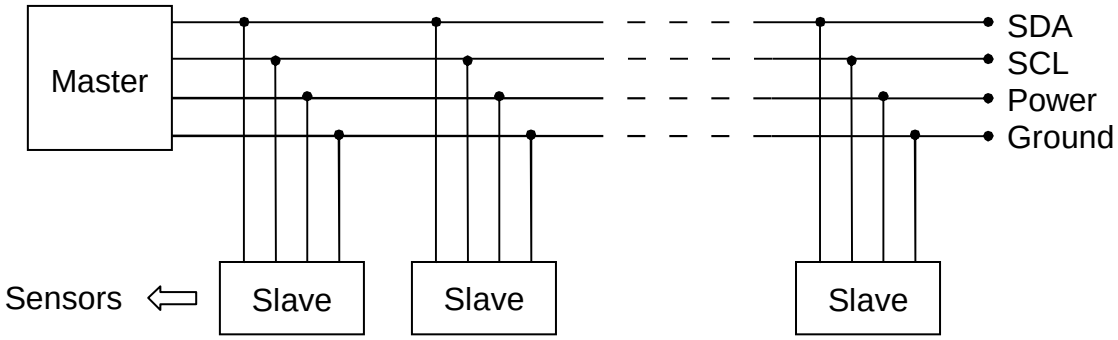

Fig.6.5: An I²C Network

I²C address of the targeted device is sent in the first byte and the least significant bit of this initial byte indicates if the master is going to send or receive data from the receiver. Each transmission sequence must begin with the *Start* condition and end with the *Stop* or *Re-Start* condition. If there are two masters on the same I²C-bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating the *Start* command at the same time. Once a master (e.g., microcontroller) has control of the bus, no other master can take control until the first master sends a *Stop* condition and places the bus in an idle state.

6.2.2 The Microcontroller

Among the many microcontrollers available which could do the interface between the sensors and the main controller, in this project OOPic-*R*, from Savage Innovations was selected. The name OOPic is an acronym for *Object-Oriented Pic*, and the abbreviation *Pic* designates a family of microcontrollers developed by Microchip Technology Inc. The *R* refers to the board layout *R*, which presents a RS232 connector.

This microcontroller supports a local I²C-Network and the capability of networking through I²C-Bus as well. In the local network, devices are appended as slaves to the microcontroller, which acts as master. In normal networking (not by the local bus), the microcontroller acts as master and slave, what is organized by an arbitration process. Those capabilities become very important because of the address limitation of the SRF08. This ultrasonic sensor reserves just

four bits for the address, making possible just 16 different addresses and therefore at most the same number of sensors at the same I²C-Network. As the sensors are to be appended to the local I²C-network, many controllers may be appended in a bigger network (up to 120) each of them having to up to 16 sensors connected to it, as shown in Fig. 6.6. Since this microcontroller has a relative low cost, the addition of extra ones does not rise the overall cost of the implementation too much; moreover, the system keeps its high modularity.

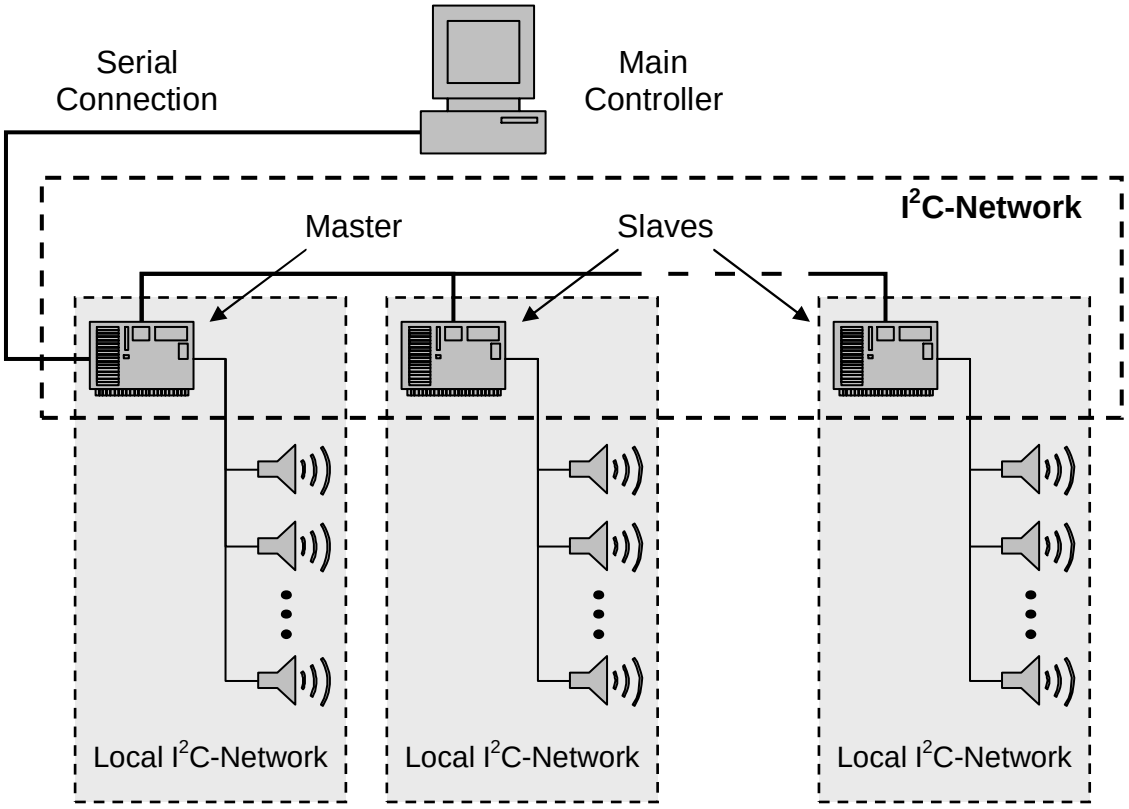

Fig. 6.6: Hardware structure

This microcontroller uses a Serial Control Protocol (SCP), what allows a device with a serial port to interact with, control and debug an application while running on the microcontroller, including directly interacting with the application's objects. Then, with help of this protocol, serial communication is used to state the connection with the PC, which is responsible for the main control of ALDURO. In this way, the information can be passed to the computer in a very simple fashion, and the microcontrollers used can be started, stopped, reset and even the sensors are possible to reach directly.

The information sent by the microcontrollers consists just on the measured value and a specification of which sensor has performed the measurement. As the time necessary to

accomplish a measurement is much longer than the time necessary to send the information to the main controller, no time references between computer and microcontroller are necessary.

The concept behind OOPic is straightforward: use preprogrammed multitasking objects from a library of optimized objects to do all the work of interacting with the hardware. Then it is just necessary to write small scripts in a high level programming language (Basic, C, or Java) to control the objects. During operation, the objects run continuously and simultaneously in the background while the scripts run in the foreground telling the objects what to do. Many aspects of the objects can be controlled by the scripts as the objects do their work with the hardware.

6.2.3 The Software Platform

The modular software environment MCA2 – Modular Control Architecture – is used for the implementation of the control system [46][87]. Its encapsulated modules are arranged in a parallel / hierarchical structure, where each module consists of two defined access ports (*sensor* and *controller*) each one with own inputs and outputs. Each module carries out a specific task and communicates with the modules, which are connected to it. In this control platform, single modules can be easily exchanged by other compatible ones.

Three modules compose the implementation of the collision avoidance system. The first one serves as interface to the serial port, reading the values sent by the microcontroller. Then these values are sent to the group *Extended Sensorik* in ALDURO's control system, where a module is responsible for the entire map building activity. The so formed map is kept in a common memory area, to be used by a module of the group *Motion Generation*, which consists on the collision avoidance itself.

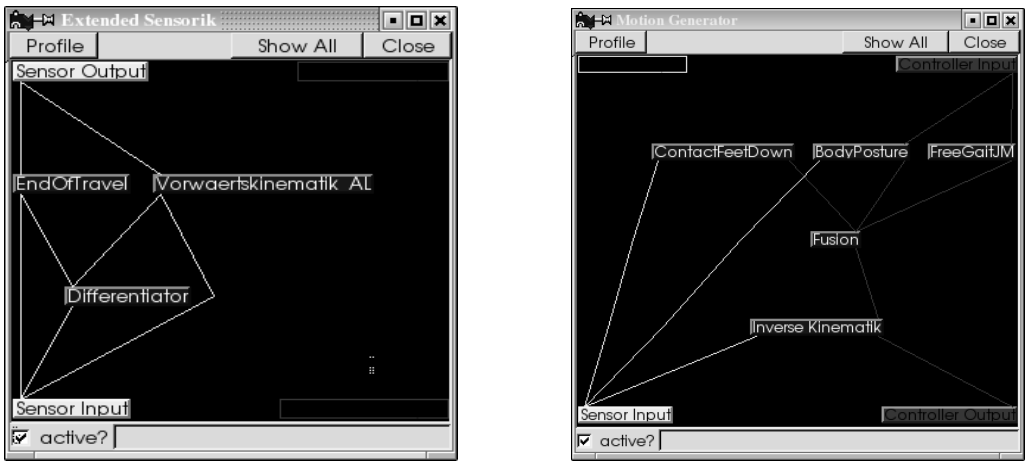

Fig. 6.7: examples of MCA2 windows

All the modules in the MCA2 are implemented using the C++, and a graphical user interface is available, where is possible to see the whole control structure or individual groups, as shown in Fig. 6.7.

6.3 Sensors' Placement

Many configurations are possible for placing the sensors over the robot [9], [10], [102], but to achieve the objective of collision avoidance; these guidelines have to be followed:

- Maximize the covered volume, trying to have a more intensive cover in the travel direction of the legs.
- Avoid cross talk, which is the reading by a sensor of echoes generated by pulses emitted by another sensor. That happens when the two sensors are pointing to the same direction and being fired almost simultaneously.
- Maximize the overlapping of sensor workspace, especially where a more intensive cover is desired, in order to improve the precision of position estimation for the objects in the surroundings.
- Minimize the number of sensors, due to economical and technical reasons. A smaller number of sensors would require a smaller number of microcontrollers too.
- Assembly constraints have to be respected, so that the attachment of sensors to the robot, as well as their orientation, does not interfere with the robot activities and is physically feasible.
- Minimize the total measurement time. This is the time necessary for a complete round, that is, for all sensors to make a measurement (if the sensors are not fired simultaneously).

Cross talk is always a serious problem with ultrasonic sensors, and not much can be done to avoid it. The simplest solution is to place the sensors so that their workspaces do not overlap, but that is not always possible (would require too much space on the robot) or desirable (one of the guidelines is the maximization of overlapping). Another solution is to fire the sensors sequentially, with a time interval between their emissions long enough to completely dissipate the energy emitted by the former sensor. That is in general feasible, but the total measurement time would be too long. A more elegant solution is the combination of a careful placement of the sensors and alternated firing, enabling to fire more than one sensor at once, enhancing then the rate of measurements. Given a configuration of the sensor array, the sensors are

distributed in groups, to state the firing order. The groups are fired sequentially and, to fire a group means to fire all sensors of this group simultaneously.

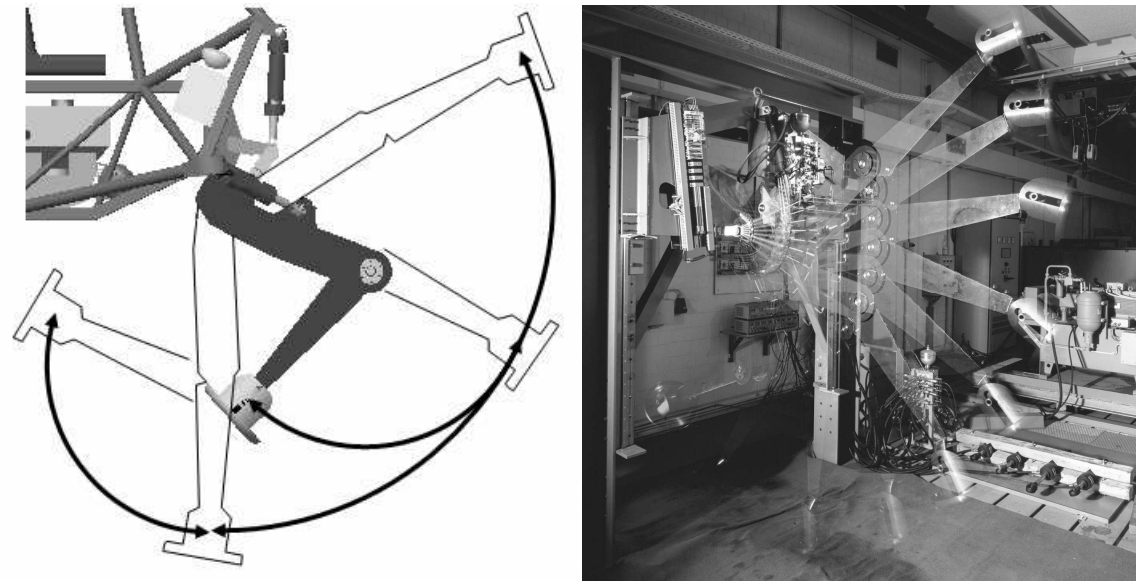

Fig. 6.8: Leg movement (left: schematic; right: leg test stand)

As the legs move, their points describe trajectories like those in Fig. 6.8. The movements of the legs will be used to move the sensors; in this way, the management of step motors or any other devices for sensor motion is avoided. Moreover, this procedure has the advantage that the sensors will be aiming at the direction to where the robot is going. If there is no deflection due to the collision avoidance system, the legs will perform a planar movement and the sensors will follow this planar trajectory, so, in order to obtain a larger cover, two groups of sensors are used at each side of the legs, with a small angle relative to the motion plane.

As ALDURO has a symmetric structure, the configuration adopted for the sensors is the same in all legs, and at both sides of each leg. The sensors at the rear legs, at first, do not collect new information; however, they are necessary if the robot, for any reason, needs to walk backwards. There are many different possibilities to place the sensors on the leg, they are:

- *Parallel over the leg*: depending on the length of the leg, two or three sensors may be placed at each part of it (thigh an shank). The emission axes remain perpendicular to the respective leg's part axis, as shown in Fig. 6.9. That enables very precise measurements of objects direct in front of the leg; however, this variant leads to relative big blind areas at the outside of the joints.

- *Inclined over the leg*: three sensors are attached to each part of the leg. The middle sensor has its emission axis perpendicular to the leg's part axis, but the two other have

their axes inclined to the joints near to them, as show in Fig. 6.10. This inclination enables a larger covered space, with smaller blind areas and keeps a good precision when measuring the distance to objects direct in front of the leg.

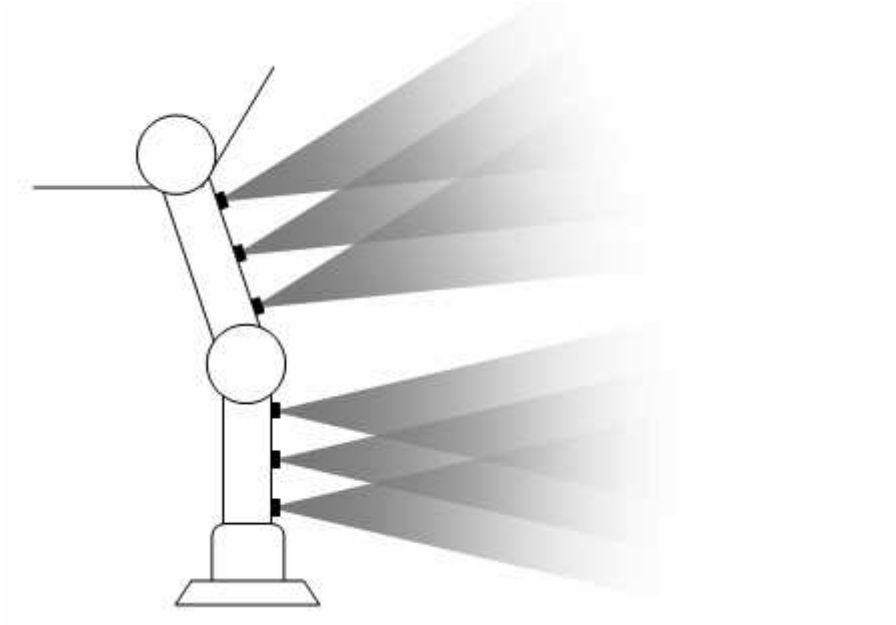

Fig. 6.9: Parallel configuration of sensor array

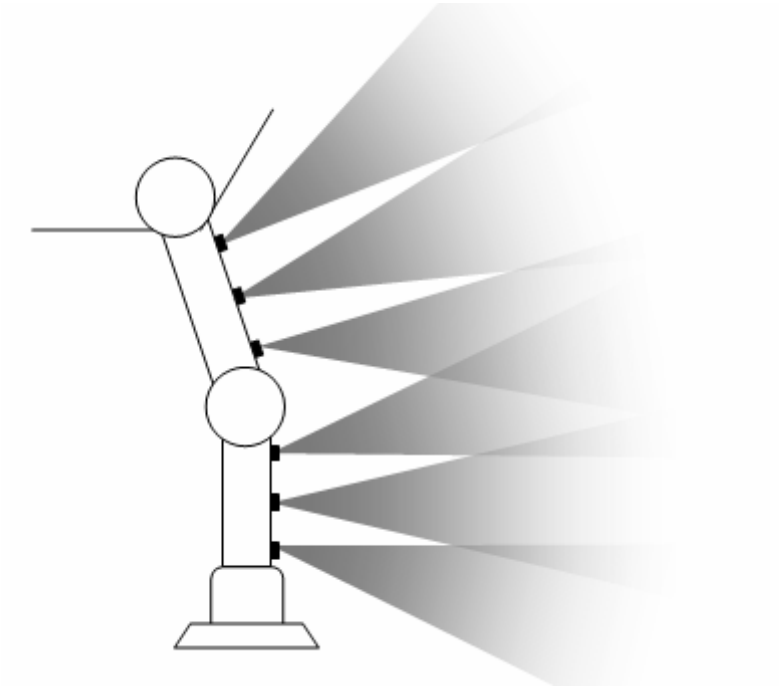

Fig. 6.10: Inclined configuration of sensor array over the leg

- *Crossed over the leg*: here also, three sensors with inclined emission axes are used, and the middle sensor has its axis perpendicular to the leg's part axis. The two lateral

sensors are inclined to the middle one, as shown in Fig. 6.10. In this configuration is possible to cover a larger volume with fewer sensors; however, just objects relatively far away from the leg can be detected.

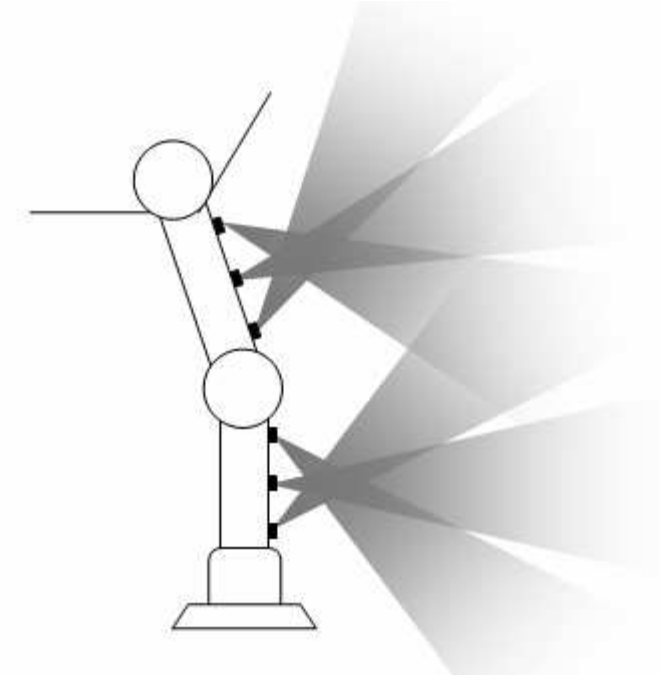

Fig. 6.11: Configuration of sensor array crossed over the leg

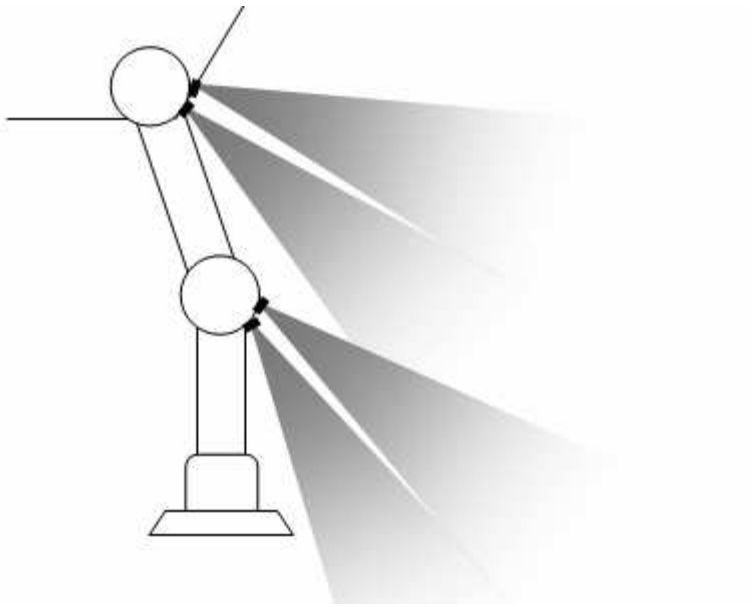

Fig. 6.12: Inclined configuration of sensor array on the joints

- *Inclined over the joints*: in this case the sensors (two or three) are not mounted on the leg parts but on the joints and their emission axis are inclined, with the beams along

the leg, as in Fig. 6.11. This configuration enables to cover a very large volume too, but the precision may be slightly degraded.

If the leg trajectory of Fig. 6.8 is considered, it is easy to see that sensors placed on the thigh will not directly cover the ground when any of the three first configurations presented here is employed. Especially in the two first configurations (*parallel* and *inclined over the leg*), if the leg is stretched, the sensors on the thigh will detect only very high objects; if the leg is bent over the body, probably no objects can be detected at all. Moreover, considering the sensors placed over the shank, the parallel configuration tends to collect information about objects just when they are still far away, being not possible to improve the estimation of positions when the objects are near. This problem is not so critical in the configuration with sensors inclined over the legs and does not exist in the crossed configuration.

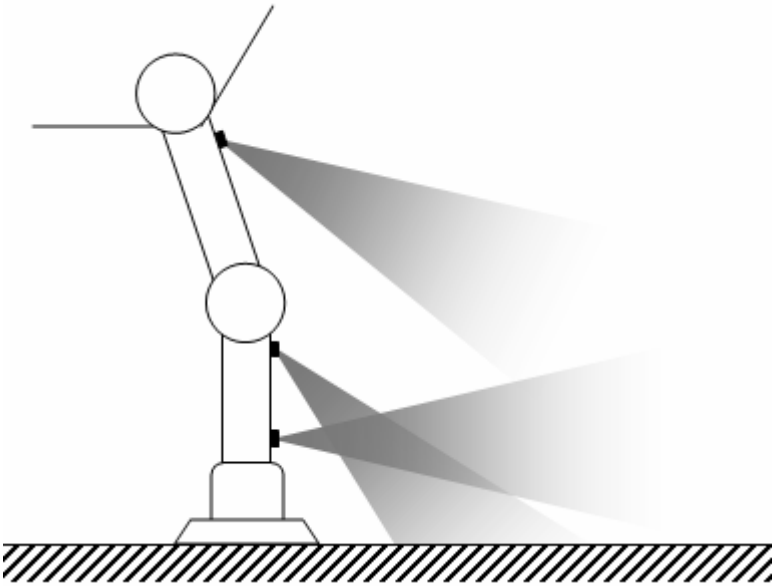

Fig. 6.13: Modified crossed configuration of sensor array

The configuration with sensors *inclined over the joints* gives a good covering of the area very near to the foot and at a middle distance too. However, in the case of ALDURO, the assembly would be problematic, mainly on the hip joint, because it is actually a composed joint. Actually, this configuration presents some similarity with the crossed one, which can be slightly modified to obtain the desired configuration. First, eliminate the sensors on the thigh pointing upwards and forwards; then, eliminate the sensor on the shank pointing upwards, as in Fig. 6.13. The remaining sensor on the thigh covers the middle distance; the sensor on the shank pointing downwards covers the close area and the one pointing forwards, objects far away.

The modified crossed configuration requires three sensors at each side of the leg, with a total number of six sensors per leg. That leads to twenty-four sensors over the robot as shown in Fig. 6.14. Therefore, two microcontrollers are necessary, which enable the placement of more eight sensors without need of employing more microcontrollers. As the sensors are relatively low cost and the whole system is modular, extra sensors may be attached over the robot body, since they do not interfere with the detection of the ones placed over the legs.

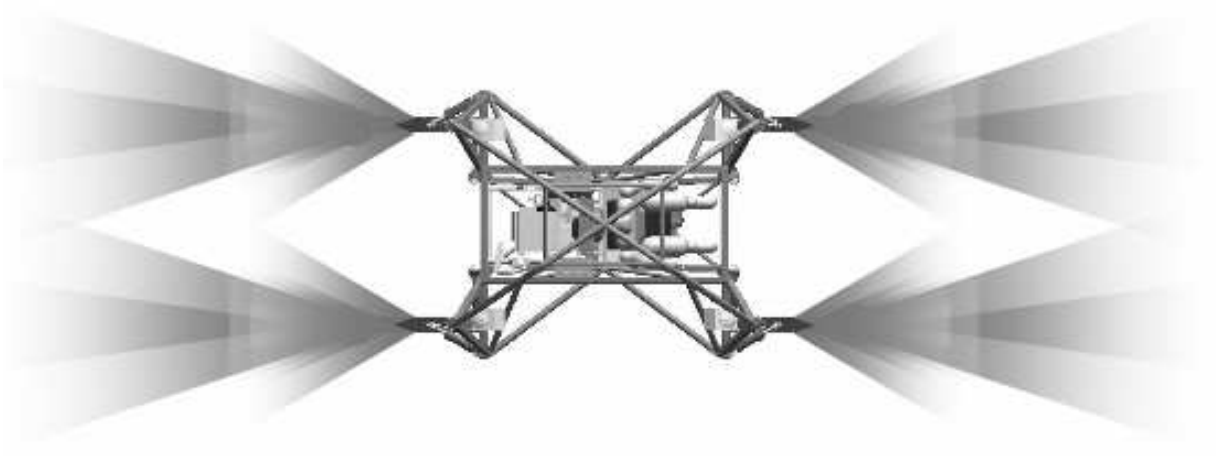

Fig. 6.14: ALDURO with ultrasonic beams

7 Tests and Results

Results of different tests are presented in this section. First, results of simulations are shown and the performances of the fusion module and of the controller are discussed. The next section brings results of tests done in a static environment, with help of a small mobile robot; but the objective was really the comparison of different combinations of partitions and output functions at the fusion module, to find the most appropriated relation of parameters. Later are presented tests done with the same mobile robot, but here the control part is analyzed too, showing the efficiency of the proposed method. Finally, are shown results of tests on the leg test stand.

7.1 Simulation Tests

A comparison is carried out, between the here proposed fusion method and a classical occupancy grid with fuzzy fusion. Models of terrain are generated in the computer to feed directly the fusion processes, so that they can be tested without interference of external disturbs and influence of the sensors. In the fusion through TSK system, two different linear functions were tested as output functions: constants and planes. To execute the simulations, the following issues were considered:

- different input terrains are randomly generated, to represent the widest range of possible terrains;
- different numbers of sample points are taken using the same computer generated terrain. In this way it is possible to state a relation between the size of the sample and the quality of the map generated by the fusion;
- The number of partitions used in the TSK system is varied so that each system with a different partition can be treated as a different system in order to compare separately the performance of each one.
- For the occupancy grid, the intersection of the complement of the ‘empty’ and the ‘occupied’ maps is used for the comparison.

First, the behavior of the fusion processes relative to the density of sample points per area is analyzed. As expected, the quality of the generated maps is improved as the density of the sample increases. This behavior is independent of the unevenness of terrain or number of

partitions used; therefore a standard density of 100 points /m^2 is used in the following simulations.

The next factor to be analyzed is the unevenness of the terrain. This unevenness is calculated by taking the *z*-coordinate as function of the *x*- and *y*-coordinates; then a numerical approximation of the gradient of this function (*z*) in relation to *x* and *y* is calculated over all points of the surface. Finally, the summation of the square of all gradients is taken and divided by the number of points. Actually some cases may be ambiguous, but in general that measure is enough to make a distinction of the different kinds of terrain.

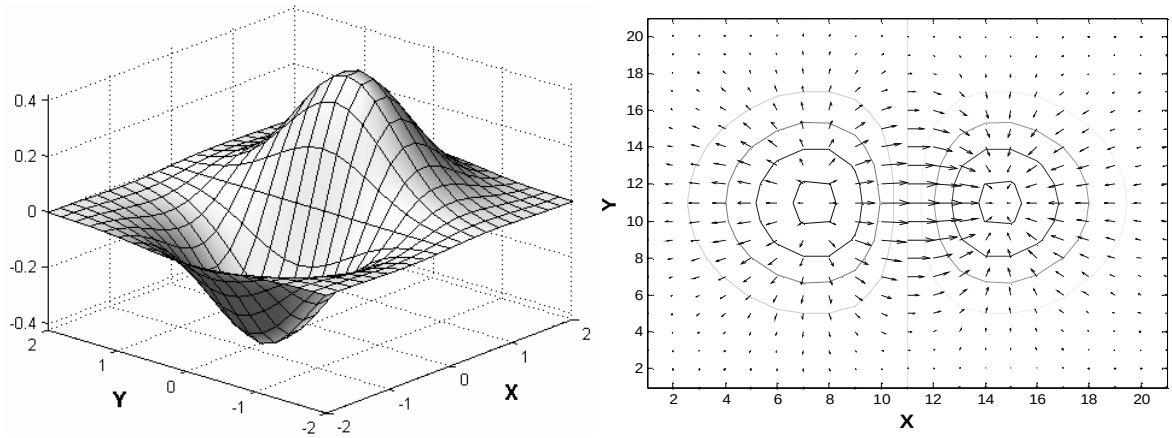

Fig. 7.1.: Generated terrain (left) and respective gradients (right)

In Fig. 7.1 a surface is shown together with its contour plots and its gradients, which are used to calculate the unevenness. Figure 7.2 shows that a larger number of partitions gives a better quality to the map; however, it is not desirable that this number increases too much, because more partitions means more parameters and therefore, a longer processing time. Besides, it is well known that smaller partitions tend to capture more noise.

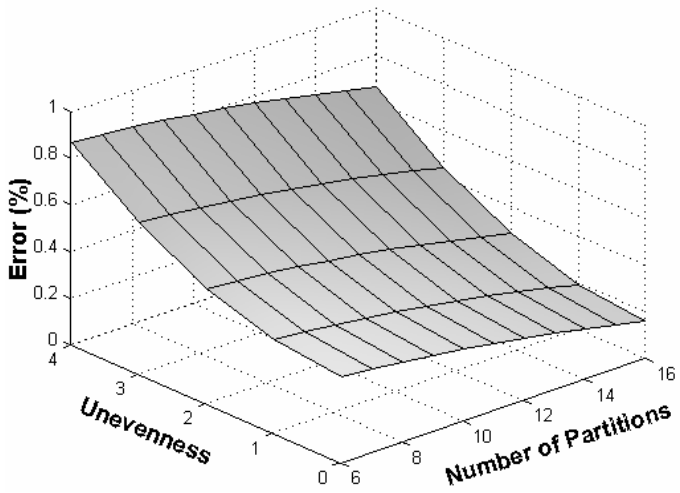

Fig. 7.2: Relation between error, unevenness and number of partitions

Fig. 7.3 shows results obtained from a representative simulation. The contour plots in this figure represent the input terrain with a unevenness of 2.4 (7.3.a), the map obtained from the classical occupancy grid using directly fuzzy logic (7.3.b), the map constructed through TSK system with constants as output and 15 partitions (7.3.c) and with planes as output functions and 8 partitions (7.3.d).

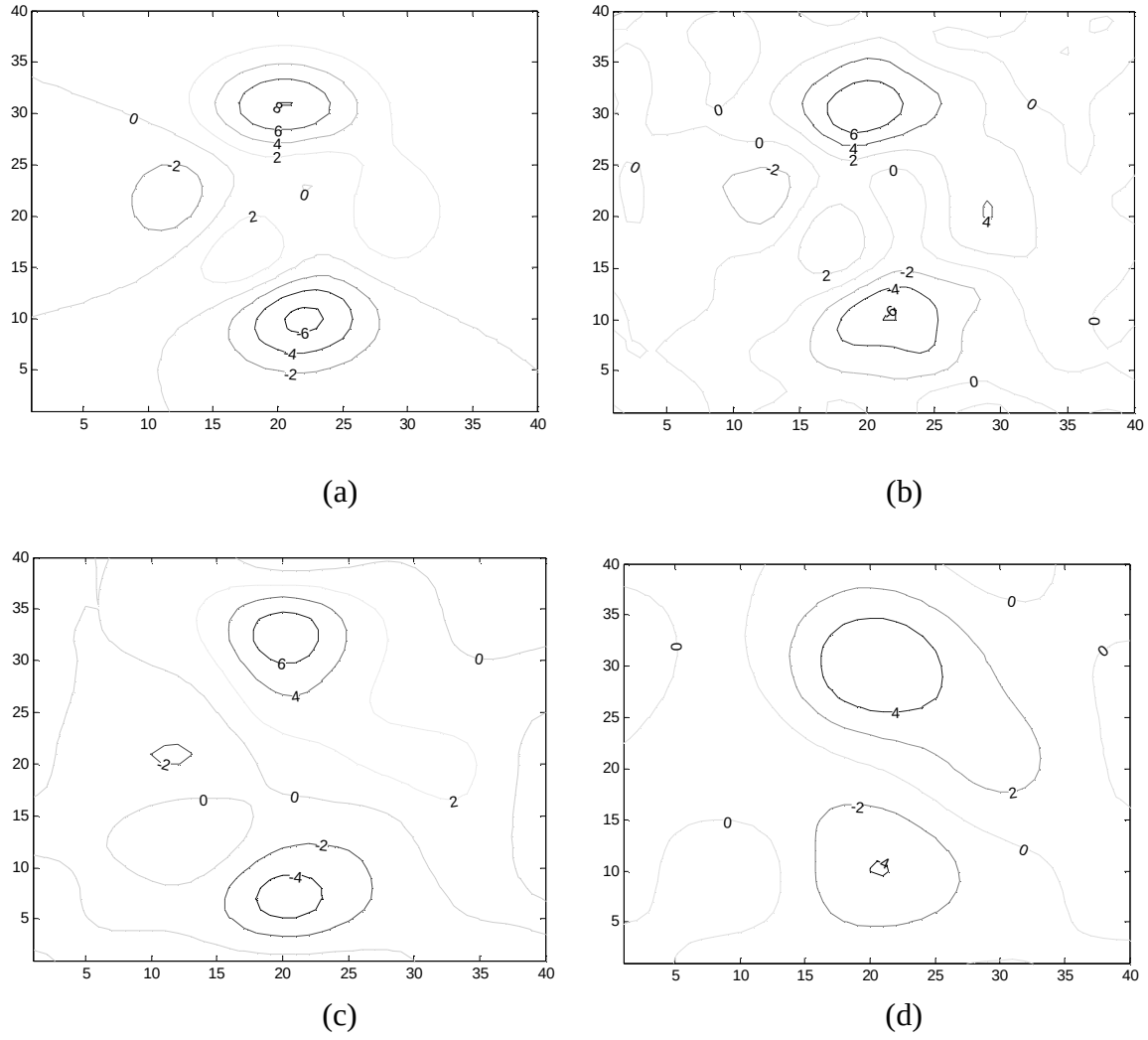

Fig. 7.3: (a) Input Terrain; Maps obtained by (b) TSK with planes as output functions; (c) TSK with constants as outputs and (d) Occupancy Grid.

As expected, the TSK system seems to be a little more sensitive to different kinds of terrain than the occupancy grid, what can be overcome by an adjustment of the number of partitions, holding then the general performance. Comparing the best results of both to a given terrain as in Fig. 7.3, the resulting map obtained by the TSK fusion matches much better the original terrain than the one obtained by occupancy grid. If the number of samples is varied, the TSK fusion gives smoother results than the occupancy grid, what is more remarkable when planes are used as output functions.

Simulations are carried on with a simple model of four legged robot, shown in Fig. 5.11, and the many input terrains generated. The sensors were modeled by their workspace, with the placement stated in Chapter 6. The overall result of controller and fusion proposed in Chapter 5 was satisfactory, since in all simulations there was no collision and in more than 80% of the cases the robot achieves the desired arrival point. A representative case can be seen in Fig.7.4, where the starting point is at (1; 5) and a desired final point is at (9; 1). The robot performs the travel without problems, reaching the target point and avoiding the possible obstacles on its way. The use of a local map has increased the processing time in comparison to pure reflexive actions, but the performance of the robot at the simulations shows that it is worth.

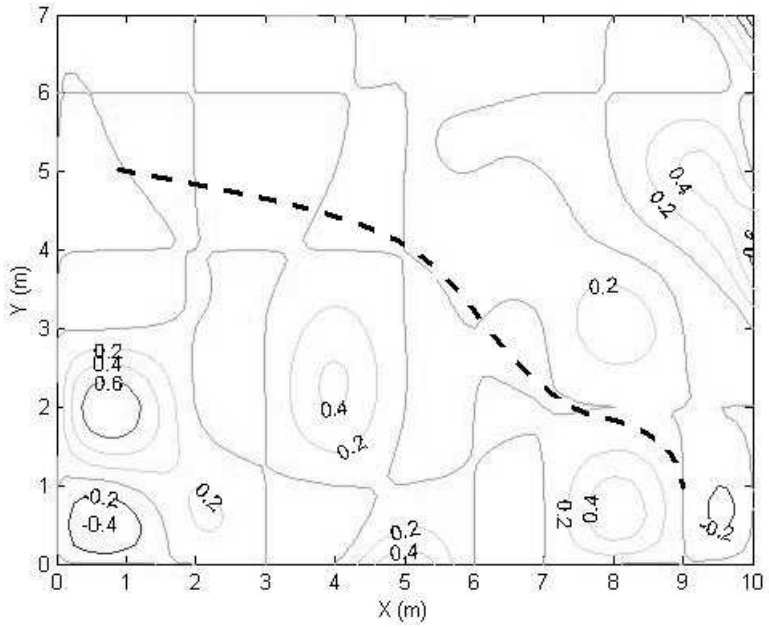

Fig. 7.4 - Robot path during simulation

7.2 Static Tests

In this section, results of the fusion using the TSK system with real sensor data are displayed. For these tests, first the data from the environment was collected and then all the processing was done off line; hereby, the set of measures used in all the tests is the same. Such tests are usually called *static* because the sensors are manually placed in different positions and measurements are done at each of these positions; but here a small mobile robot, to be introduced in next section, was used to do this positioning of the sensors. This robot carries four sensors in the same configuration to be used on the ALDURO's shanks (see Section 6.3).

The environment to be recognized consists on two simple objects shown in Fig. 7.5 (left), one of cylindrical form, and one of prismatic form. The area to be detected is 60cm×60cm, and

robot makes a complete circle around the target area, with the sensors pointing to its center, with just one firing per time, to avoid cross talk. A solid model of the environment, shown in Fig. 7.5 (right) was constructed to enable the comparison of obtained maps after fusion and to calculate the error.

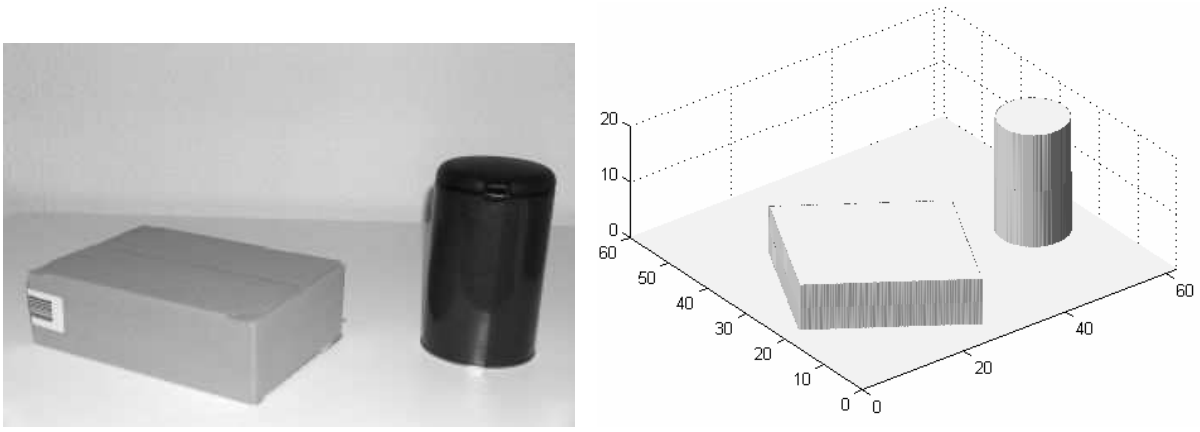

Fig. 7.5: Environment (left) and its solid model (right)

As already mentioned, the presented fusion method is sensible to the number of partitions used. Zero order output functions were used to make the estimations and the error between the obtained map and the solid model can be seen in Fig. 7.6. There is shown the relation between number of partitions and error, which decays with the increase of the first. The error is calculated similarly to the *RMS* error as $err = \sqrt{((z - \hat{z})/z)^2 / n}$, where n is the number of samples.

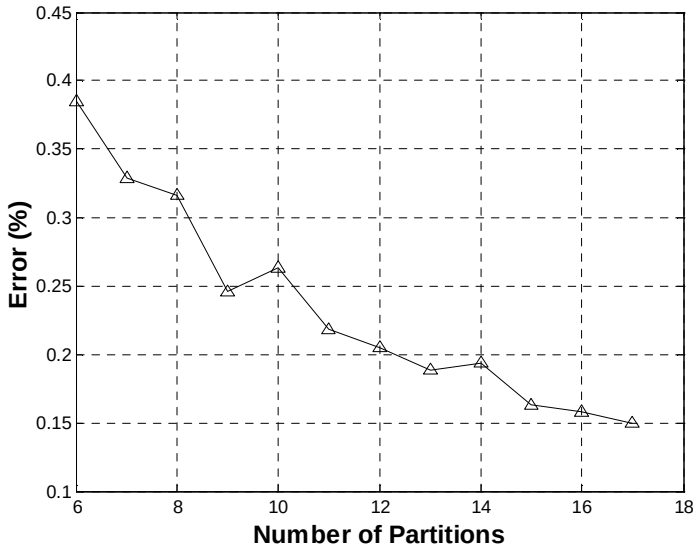

Fig. 7.6: Error with different partitions for zero order functions

It is interesting to see in Fig. 7.6 that the curve is not smooth, and in spite of having larger number of partitions, some configurations have higher error than other with fewer partitions. That is because of the geometrical adjustment of the fuzzy partitions to the 'form' of the system to be identified. Neverthenless the error decreases as the number of partitions increases.

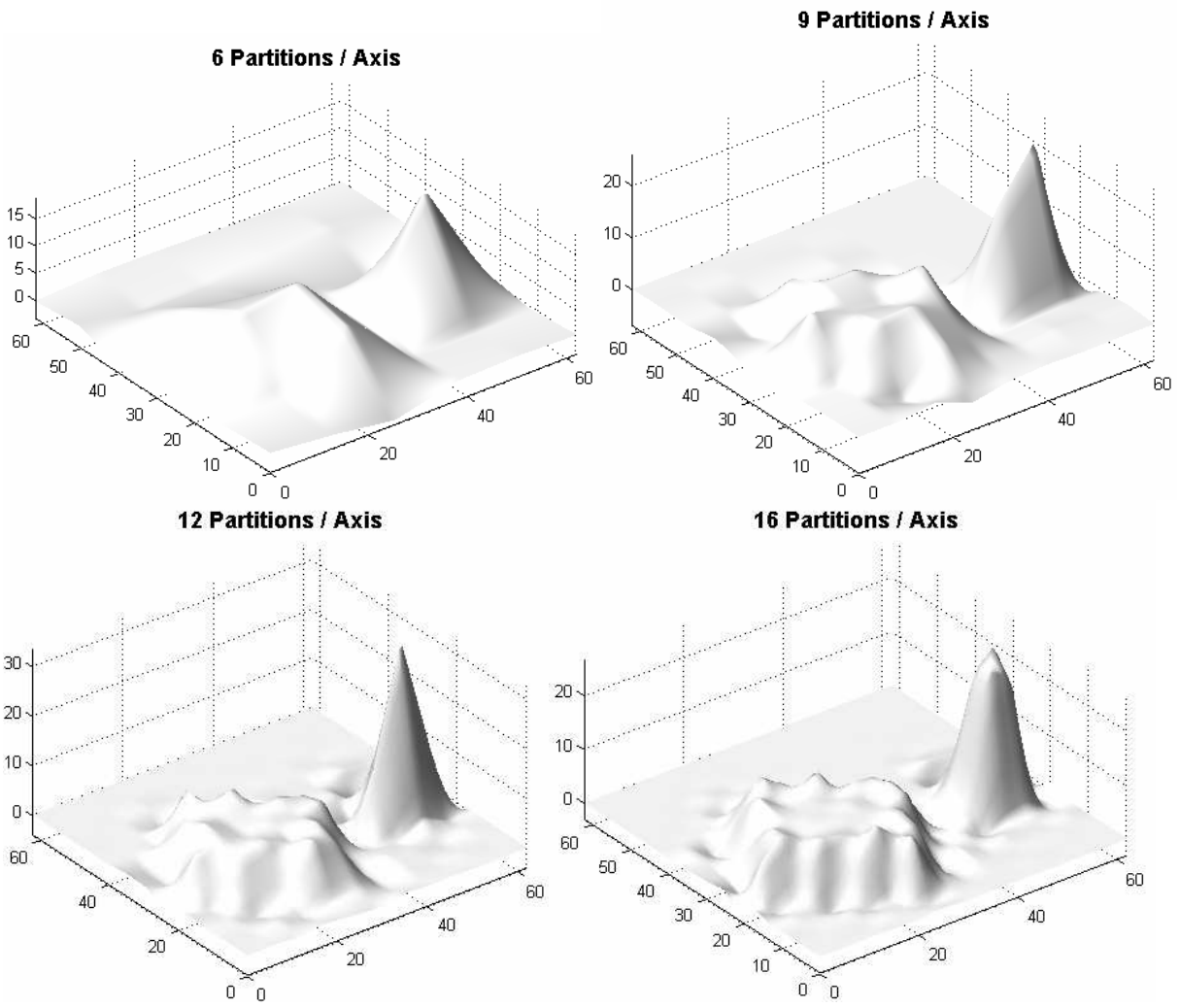

Fig. 7.7: Estimation with zero order functions

In Fig. 7.7, some of the maps estimated with zero order output functions are shown. The quality achieved with sixteen partitions at each axis and 256 parameters to be estimated is acceptable for navigation. Besides the map quality, also the processing time is significant for the practical application of the method, as the robot has to process all the data in real time. In Fig. 7.8, the relation between number of partitions and processing time for one measured value is shown. This time increases approximately square of the number of partitions per axis, what means that it increases linearly with the number of parameters.

As the measurements are processed sequentially, the maximal time available to process one measure is the time until the next measure to come. As the sensor takes 65ms for one measurement one can process maximally sixteen partitions per axis with processor AMD Athlon 2800^{TM}.

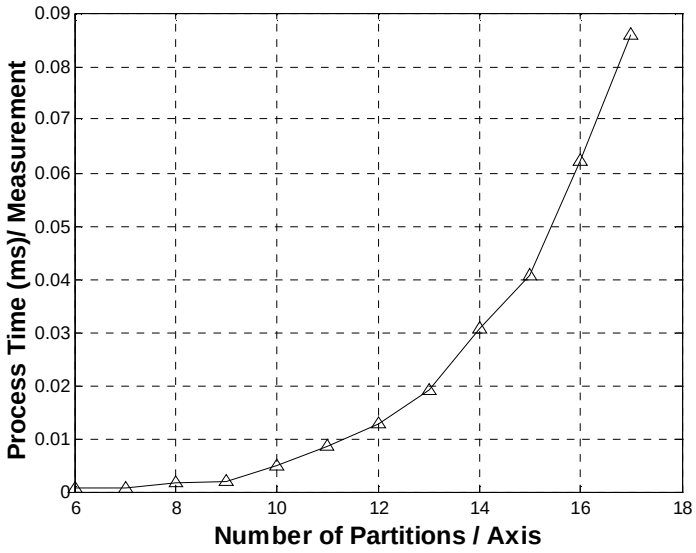

Fig. 7.8: Process Time with different partitions for zero order functions

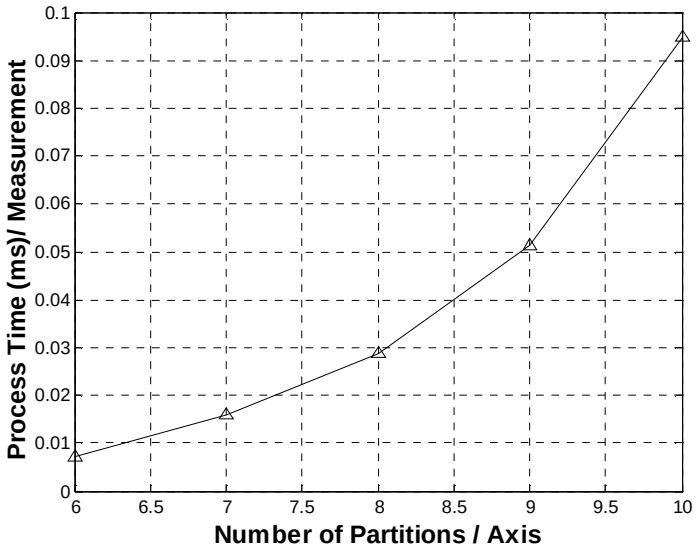

Fig. 7.9: Process Time with different partitions for first order functions

The same curve of time versus number of partitions is traced in Fig. 7.9 for estimations using first order functions as output functions. It is possible to see that it grows with the square of the number of partitions, but much faster than in Fig. 7.8. That is because in first order output functions three parameters are used, what would lead to a growth three times faster than with

zero order functions; but because of new matrices that need to be calculated (see Eq. 5.31) this growth is quadratic. Therefore, with first order functions, the time increases nine times faster than with zero order. That limits the feasible number of partitions per axis to nine.

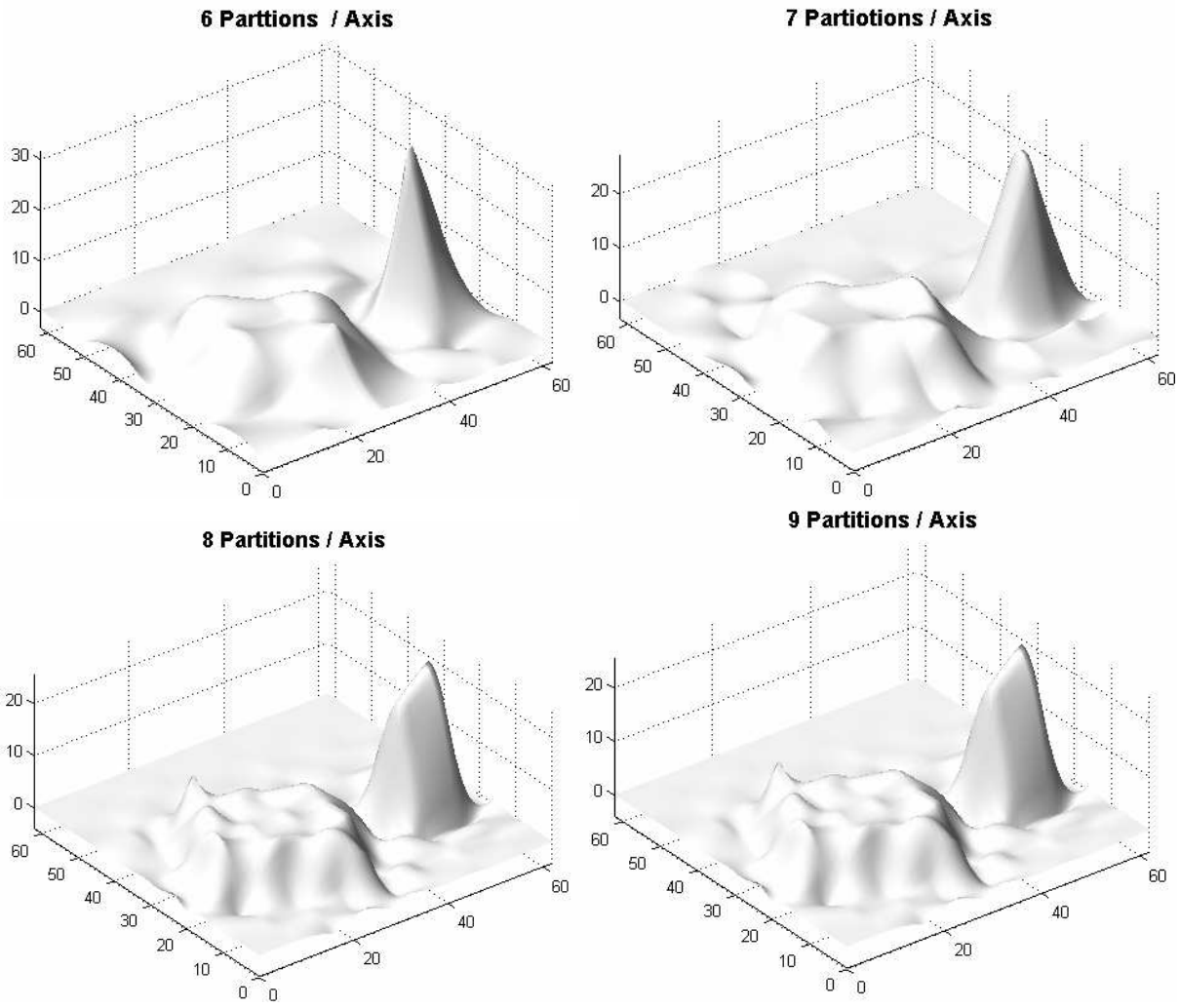

Fig. 7.10: Estimation with first order functions

In Fig. 7.10 there are shown estimations obtained with some time-feasible configurations of the system with first order functions. As can be seen in Fig. 7.11, the error for nine partitions / axis is just a little longer than for sixteen partitions /axis with zero order functions, and both of them are beyond the time limit of 65ms. The use of zero order functions has the advantage of being simpler to implement, besides it can better characterize edges. On the other hand, systems with first order functions give better results when the sample is small. Since the sample rate of the whole sensor system is high enough to provide an adequate density of sample points (10 points/m), the zero order systems should be adopted in next tests.

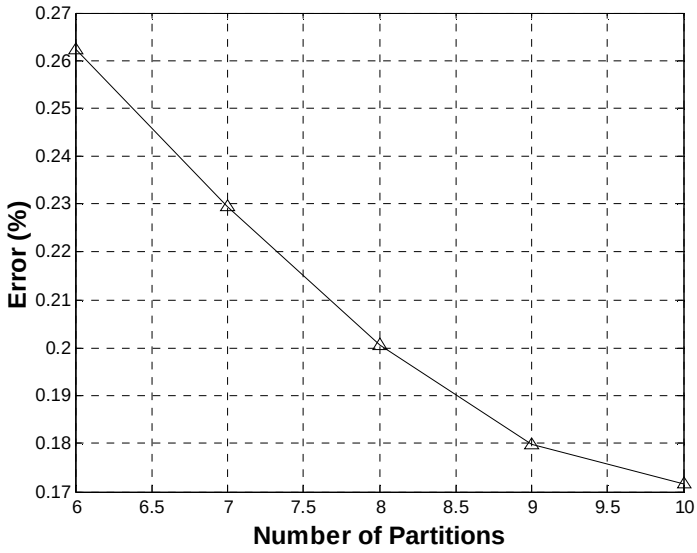

Fig. 7.11: Error with different partitions for first order functions

7.3 Tests with Small Mobile Robot

The collision avoidance system was tested in a small didactic mobile robot called Rug Warrior (Fig. 7.12). This robot consists of a body provided of two motors, which are directly assembled to the two driving wheels, and a third passive wheel has the support function. As a wheeled robot with such a configuration (Fig. 7.12 - right), Rug Warrior can only perform movements over surfaces, that is, locally planar movement. As no vertical movement is possible, a simplified version of the controller of the collision avoidance system was employed to support just planar navigation. Zero order output functions were used and a grid of $1m^2$ around the robot was defined to build the local map.

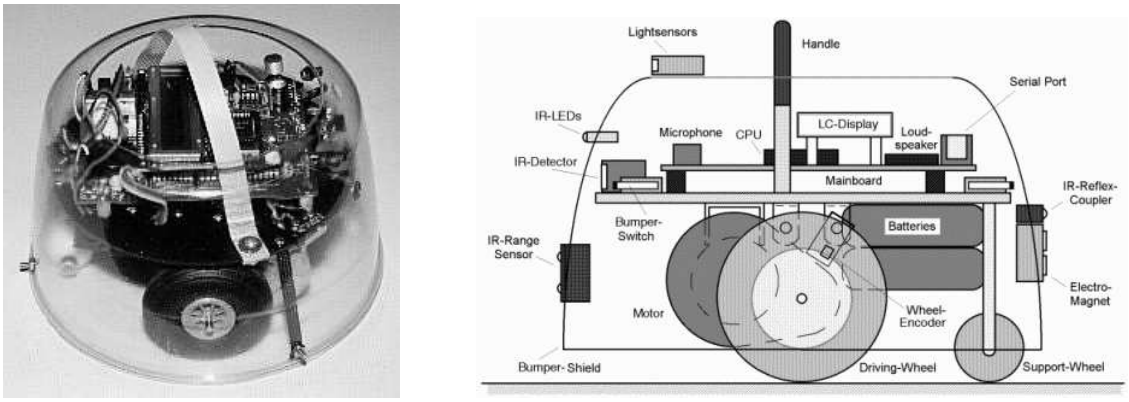

Fig. 7.12: Rug Warrior

A hexagonal platform with six ultrasonic sensors was constructed and assembled over the robot as shown in Fig. 7.13. The sensors are connected to a microcontroller on the platform, which process all the collected information, performing the collision avoidance's calculations. This microcontroller sends the corrective velocity to the robot in form of its two components, through two analogical input lines.

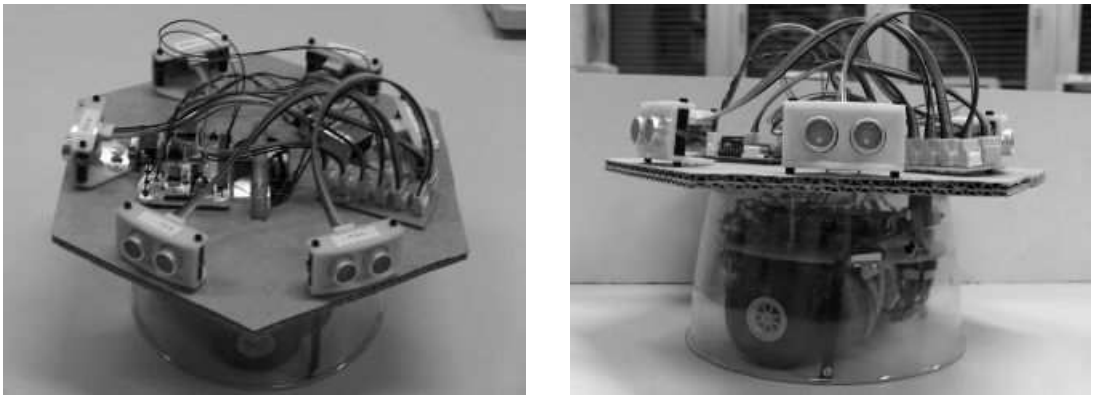

Fig. 7.13: Rug Warrior with ultrasonic sensors

The robot can be programmed from a PC through the serial port using *Iterative C* language (a simplification of the *C* language). Then, for the first test the robot was programmed to find the free way and to follow it. A labyrinth as shown in Fig. 7.14 was build in the laboratory, the robot was placed at the entrance and should go ahead, to cross the labyrinth. As there was no way out of the labyrinth, the robot interprets the way back as the free way and comes back to the start point. It is important to note that the path followed by the robot does not present too much oscillation, especially at the end of the labyrinth, where the robot turns back. That is the main advantage of having a local map available; it gives to the robot knowledge enough about the surroundings to avoid such trapping situations.

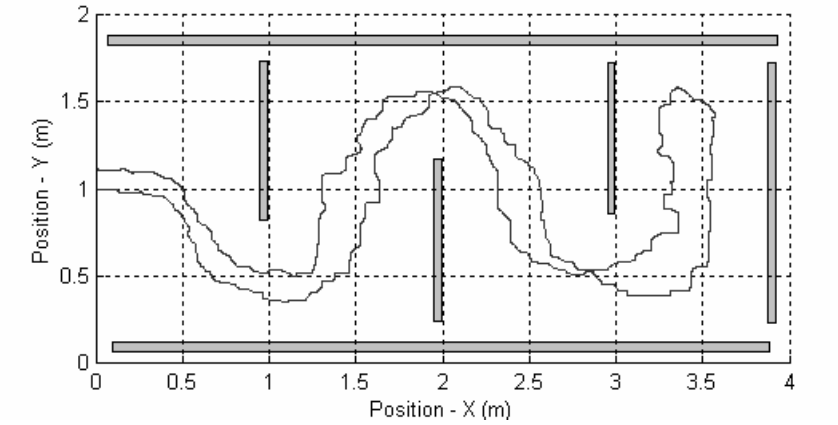

Fig. 7.14: Finding the free way

The path of Rug Warrior in Fig. 7.14 is not smooth, but that is due to the poor velocity control implemented in it. However, due to the fact the collision avoidance controller works partially as reflex action, it was possible the correction of the robot's path. Due to this poor controller, other tests in open areas could not be carried on. Other tests in relative narrow passages as in Fig. 7.14 were performed with success.

7.4 Tests with the Leg-Test-Stand

The collision avoidance system was implemented at the Leg-Test-Stand available at the Duisburg-Essen University. This test-stand consists of a leg of ALDURO in real size, with the corresponding hip joint attached to a vertically movable part. The leg is fully operating, with its four degrees of freedom actuated b hydraulic cylinders, provided of servo-valves, which are controlled by a PC.

In this computer, the control system of the test-stand is implemented based on the *Modular Control Architecture* (*MCA2*) software that provides a platform for control systems. Then, a module for the collision avoidance system it was implemented on it, leading to the structure shown in Fig 7.15.

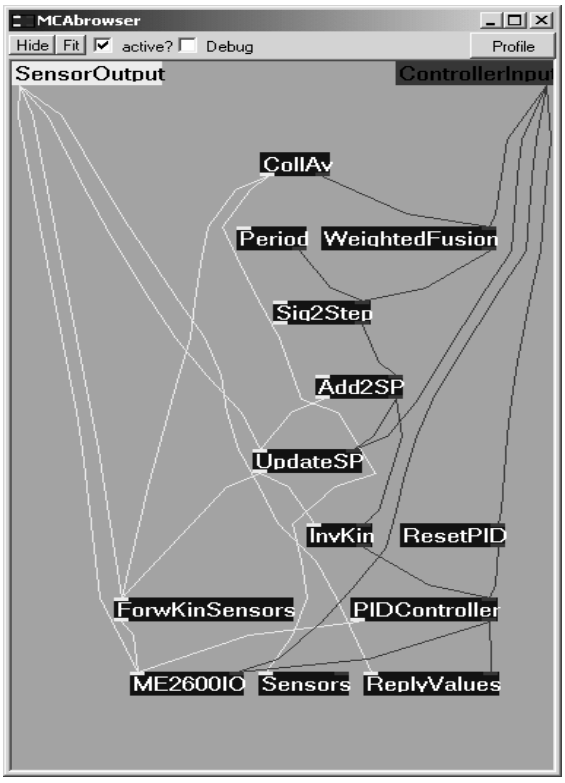

Fig. 7.15: Implementation at the control platform MCA2

As already mentioned in Chapter 6, a microcontroller was employed to realize the interface with the sensors and their management. A control unit as shown in Fig. 7.16 was constructed, containing the microcontroller and buffers that enable the use of longer lines between the microcontroller and the sensors. Besides, the buffers provide an improvement in the noise rejection.

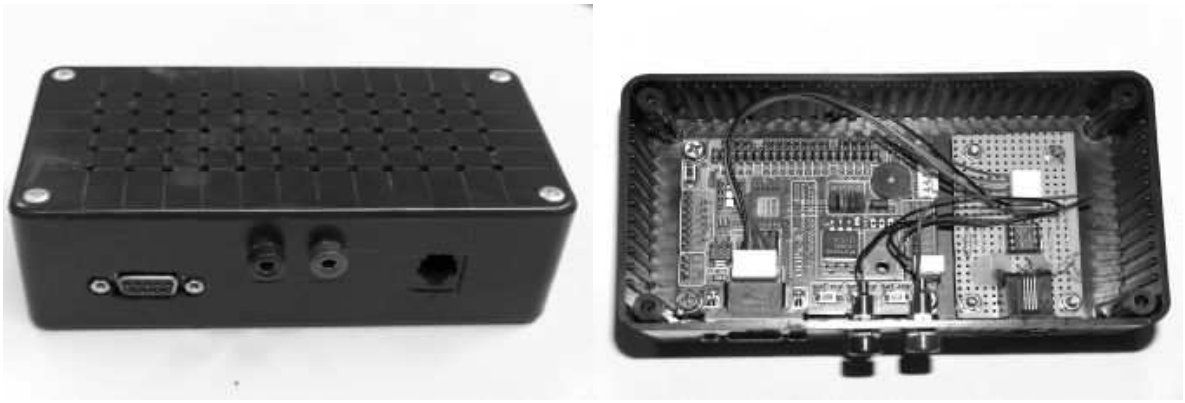

Fig. 7.16: Completes control-unit (left) and opened (right)

Moreover, an array of sensors, assembled in a parallel configuration was constructed. This realization (in Fig. 7.17 - left) permits to use less cable and enables a high interchangeability of the sensors. The sensors are placed at both sides of the leg as specified in Section 6.3. This placement consists of three pairs, where one is on the thigh and two are on the shank. Each sensor is placed directly opposed to its pair, at the opposed sides of the leg, as shown in Fig. 7.17 (right). This realization results in a relative large intersection of the workspaces of both sensors of the same pair, what is interesting, because it increases the redundant information. That enables a higher certainty degree to the points comprehended in this intersection, which coincides with the normal direction of travel of the leg.

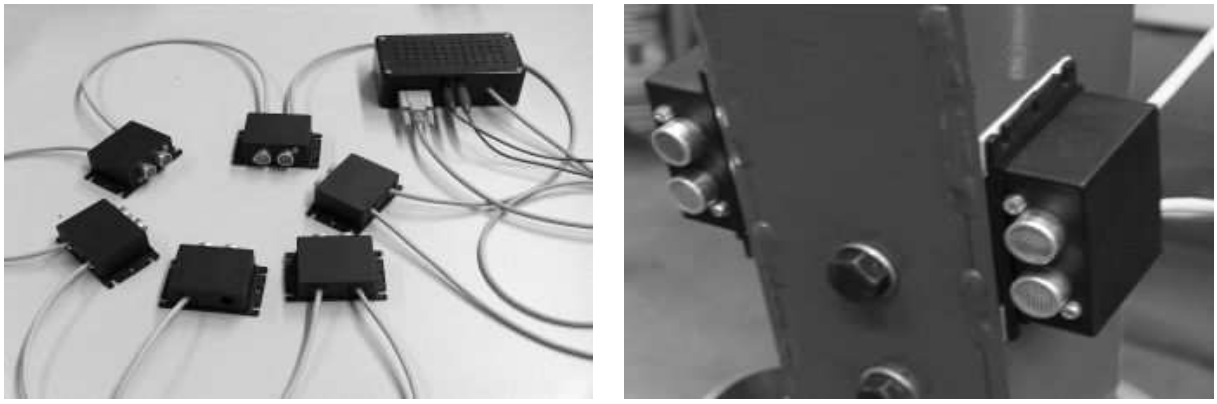

Fig. 7.17: Set of sensors and controller (left) and sensors fixed at leg (right)

In Fig. 7.18 it is possible to see the sensors placed over the leg. It is important to note that the sensors are fixed as near as possible to the frontal edges of the leg. That prevents false detections caused by adjacent surfaces, what practically may block the sensor. Long cables were employed to make possible the change of position of the sensors over the leg as part of experimentations, but that should be avoided when implementing on ALDURO.

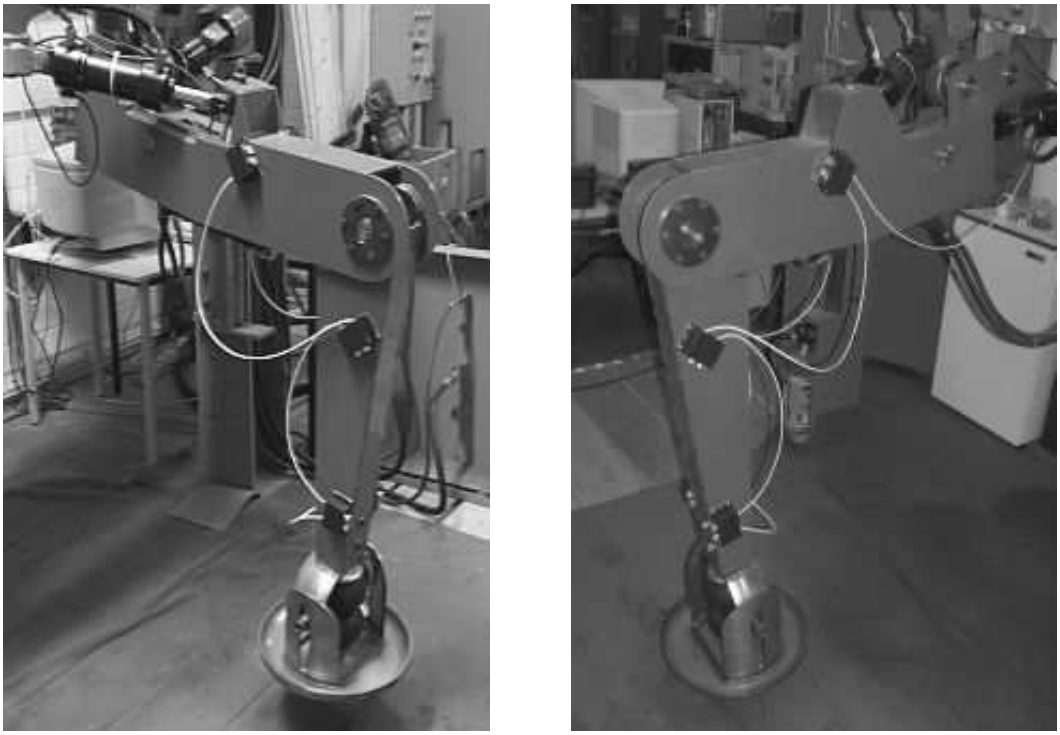

Fig. 7.18: Sensors placed at both sides of the leg

With the collision avoidance system implemented at the test stand (with software and hardware), some basic tests were performed. A constant reference velocity in the *X*-direction is imposed to the foot in three different situations, where the collision avoidance system should present a very clear response. The reference frame at the test stand are taken there as:

- Origin at the spherical hip joint
- *X*-axis perpendicular to the hip plane, pointing forwards (the side where the leg hangs)
- *Y*-axis parallel to the hip plane and horizontal
- *Z*-axis parallel to the hip plane and vertical

In the first test, a very large object, not possible to overcome, is placed in front of the robotic leg, as shown in Fig. 7.19. As the foot moves forwards, it faces the obstacle and, as expected,

it stops. There is a small deviation upwards before to stop. That is the result of the collision avoidance system trying to find another way without success.

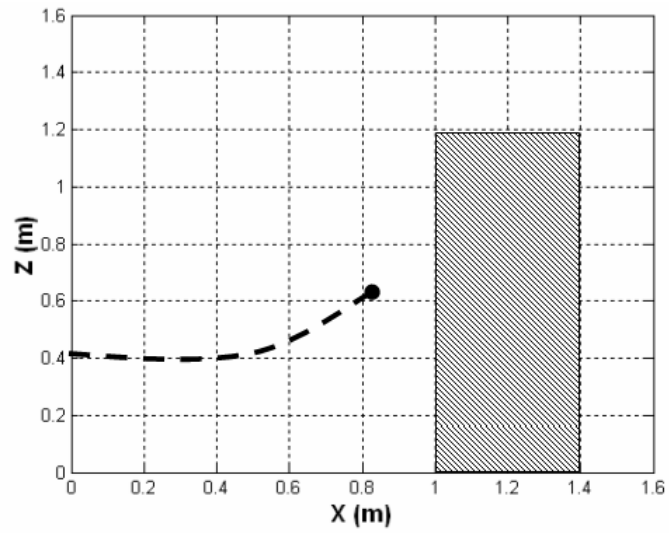

Fig. 7.19: Foot trajectory in front of a high and large obstacle

For the second test, a large object is placed in front of the leg again, but now the obstacle is possible to overcome. As shown in Fig. 7.20, as the foot tries to move forwards, the sensors detect the obstacle and the reaction of the collision avoidance system pushes the foot up, to avoid the collision.

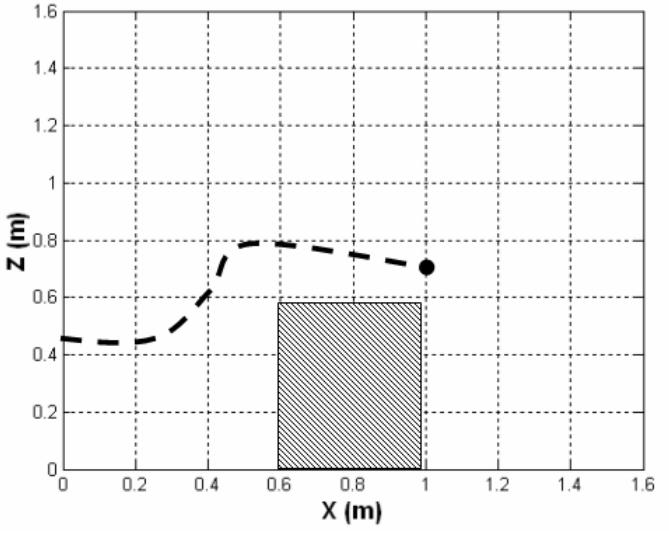

Fig. 7.20: Foot trajectory in front of a short and large obstacle

In the first two tests the obstacles were relative large and were placed symmetrically relative to the symmetry plane of the leg, in order to do not generate any lateral deviation, that is, no movement in the *Y*-direction. In this last test, the situation is exactly the opposite: lateral movement should be generated. A relative slender object is placed in front of the leg but with some offset from the leg's symmetry plane. As shown in Fig. 7.21, the foot moves forward as imposed by the reference velocity but it presents a lateral deviation.

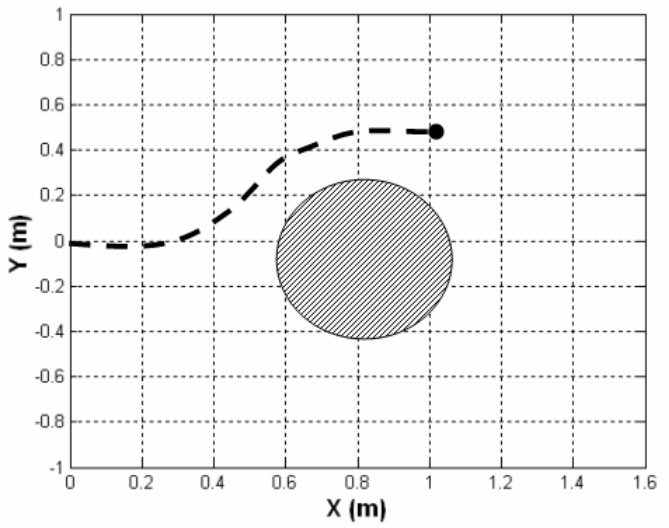

Fig. 7.21: Foot trajectory in front of a relative thin obstacle

The three tests presented here required the execution by the collision avoidance system of the three basic actions that can be executed during the movement of the leg: lateral deviation, vertical deviation (to overcome) and stop. That shows that the system works satisfactorily, being able to perform the basic actions of collision avoidance; therefore, it is able to provide a collision-free travel to the leg in more complicated situations.

8 Conclusions and Future Works

8.1 Conclusions

The objective of this work was the development of a collision avoidance system for the robot ALDURO. After careful analyses of many aspects, ultrasonic sensors were considered the most appropriated to the implementation of such system; then, a new method for real-time map building and navigation in unknown unstructured environments has been presented. The method was tailored for the robot ALDURO, but it can be easily adapted for application in other robot's projects, with high efficiency if the robot has characteristics of size and velocity similar to the ones of ALDURO. As the method was developed aiming at the application in quadruped robot, it is pretty suited for legged robots in general or robots with many moveable parts (e.g. robotic arms).

The basic feature of the presented method is the alternately execution of two fundamental processes: map building (a typical deliberative activity) and navigation (with a reactive design). In the former, the robot collects information about the surrounding scene through its sensors and updates the local representation of the environment so far reconstructed. In the latter, a suitable algorithm proposes corrections to the velocity of the feet, avoiding collisions in the explored region. The correction algorithm was designed for a robot whose motion is calculated based on reference velocities, but if that is not the case (e.g. based on force), it is necessary just to adjust the output parameters of the fuzzy controller. Besides, any part of the robot may be considered when avoiding collisions, what enables to fit the technique to be employed at robots with different structures. In the same way, so many parts of the robot may be considered as desired; the transmission of the many velocity corrections is done through the kinematics of the robot. Therefore, it is clear that the algorithm is flexible and possible to be efficiently used in many different robots (especially the legged ones).

Given the intrinsic uncertainty about the ultrasonic measurements, fuzzy set operators are used to process the measures and to update the map. The whole process of fusion is performed with employment of fuzzy logic as a tool, but its result (the local map) is a crisp elevations map, what makes possible to employ this map building process together with most of path planning algorithms available (which use crisp representations).

Experimental and simulation results have been reported to illustrate the satisfactory performance of the proposed technique. The maps obtained in simulations and static tests were quite accurate, even in environments of high reflectivity as in the laboratory for the tests with the Rug Warrior. It shows that the combination of the TSK system and the LSE method

leads to a estimator, where the effect of noise (failure reflections due to large incidence angles and high reflective or absorbing material) and of cross talk could be minimized.

Because of the kind of fusion used, old information about the environment is kept until it is far enough, falling outside of the map. That gives a certain ‘memory’ to the robot, eliminating problems of oscillating trajectories when traveling through a narrow passage. This memory is partially responsible for the robustness of the system. Besides, the paths followed by the small mobile robot in laboratory are in general safe and effective; the robot travels without problems, reaching the target point and avoiding the possible obstacles on its way.

Two kinds of linear functions were tested as output functions: constants and planes. The first one has one parameter per cell of the map grid while the second has three parameters. That makes the simulations much slower when using planes, since the dimensions of the matrices employed at the calculation are directly proportional to the number of parameters per rule. Planes as output functions have a smoother result than constants, having a superior performance when the number of samples is small. However, as the main controller of ALDURO has a high processing capacity, it is possible to use a relative high sampling frequency and then to obtain a smooth map with linear functions of one parameter and with higher reliability.

Simulations were run with different number of cells in the map grid and different input terrain. As expected the system seems to be a little sensitive to different combinations of terrain and number of partitions; very uneven terrains need a more refined grid, while terrains that are more flat can be well represented with a gross mesh. Then, the size of the cells it was taken in a way to assure the security of the robot, leading to a relative number of cells but not excessive, and the general performance of the collision avoidance is held in acceptable levels for simulations with a wide variety of terrains.

The generated map is an elevation map, represented as a continuous surface, which function of the planar coordinates. In spite of the simplicity of this representation, it has a drawback, especially for small robots traveling indoors: covered passages may not be identified in the map. If the upper part (the cover) of the passage is detected, then the passage itself will be considered as a solid object. For ALDURO that is not a problem, because as result of maximal range set for sensors and their emission directions, considered the height of the robot, the composed workspace of the sensors is not higher than that; since a possible passage has to be higher than the robot, its upper part is not detected.

These results are confirmed by the good performance of the leg in the test stand. In a environment provided with obstacles, the robotic leg could travel and avoid collisions, confirming the simulations. That partially confirms the simulations carried on for the whole

robot, which have given satisfactory results; the robot was able to travel without collision, in different terrains and to reach its goal.

8.2 Future Works

In the future, the method formulated for the walking machine ALDURO needs to be fully applied to the real robot. That is of special interest for the set up of parameters of the fuzzification, because this module is directly related to the characteristics of the hardware. ALDURO does not have a path generator, which plans long paths for the legs, instead a free gait module analyses the momentary state of the robot to generate the next movement. Since the sensors are moved together with the legs, it is important to analyze the performance of the whole real system, to assure paths given by the free gait will lead to a satisfactory cover of the surroundings, as occurs in the simulations.

For further development of the method presented in this work and its implementation, the following topics would be of special interest:

- Search for new membership functions for the fuzzification of the measured value, in order to simplify the equations obtained for the fuzzy sets, which represent the membership values of the points of the grid in local map coordinates. Another possibility to obtain such simplification is a further search on the fuzzy operators (or even to develop a new one) to make the calculations more direct.

- Search for input functions of the TSK system with overlap, for which the intersection with the fuzzfied measurement gives equations possible to solve analytically, without numerical methods (in order to keep the system as real-time). This overlap would tend to spread out the influence of the each point over the entire map.

- The introduction of secondary maps as different layers, in order to be able to represent more than one point of an object at the same *xy*-position and different heights. Two or more surfaces (just like the map here developed) could be used to represent the highest and the lowest parts of the environment. Instead of using the supremum to project the fuzzyfied measure on the XY-plane, all the local maxima of this function for each point of the grid would be considered. That would avoid the use of a three-dimensional grid (saving memory) but still would give a real representation of objects in space and not only their maximal heights.

- Development of a learning system, able to state in real time the most appropriated cell size to be used for each terrain. Such system could optimize the use of memory, providing always the necessary refinement of the grid to capture all objects with high

precision. That could be done by the constant observation of the relation between the size of forms contained in the map and the size of cells.

- The topic above leads in a natural way to the use of classification of the forms contained in the map into primitive forms (cylinders, cubes, etc…). For the navigation task that is not necessary, but could provide more details about the environment for other tasks.

- Use of cameras on ALDURO. The video data could be fused to the data originated at the ultrasonic sensors to give real three-dimensional images. The map generated by the method here presented is already in an appropriated form to be fused to video data. Such an implementation would enable the use of a safe remote-operation of the robot, where the operator in a safe operation module would have a complete spatial scene of the environment (possible too much hostile for a human being).

- Development of an arbitration process between the operator and the collision avoidance system, possibly provided of a user interface to show to the operator the environment as perceived by the robot, enabling the operator to carry an action on, even though the collision avoidance module tries to stop it.

The central point of this work was the development of a new method for real-time map building and navigation in unknown unstructured.

Appendix

A Technical Data of the Hardware

A.1 SRF08

The default shipped I^2C address of the SRF08 is 0xE0, in addition to the above addresses, all sonar's on the bus will respond to address 0 (the general broadcast address). This means that writing a ranging command to I^2C address 0 (0x00) will start all sonar ranging at the same time, but results must be read individually from the real address of each one. This sonar is always a slave, never a bus master.

Registers and Commands

The SRF08 appears as a set of 36 registers, as shown in Table A.1. Only locations 0, 1 and 2 can be written to. Location 0 is the command register; it is used to start a ranging session and cannot be read. Trying to read from location 0 returns the software revision.

Table A.1: Sensor Registers

Location	*Read*	*Write*
0	Software Revision	Command Register
1	Light Sensor	Max Gain Register (default 31)
2	1st Echo High Byte	Range Register (default 255)
3	1st Echo Low Byte	N/A
…	…	…
34	17th Echo High Byte	N/A
35	17th Echo Low Byte	N/A

Location 1 is the onboard light sensor. Locations 2 and 3 are the 16 bit unsigned result from the latest ranging (high byte first). A value of zero indicates that no objects were detected. There are up to a further 16 results indicating echoes from more distant objects.

There are three commands to initiate a ranging (80 to 82), which return the result in inches, centimeters or microseconds, shown in Table A.2. There is also an ANN mode (Artificial Neural Network) and a set of commands to change the I^2C address.

Table A.2: Sensor commands

Command		*Action*
Decimal	*Hex*	
80	0x50	Ranging Mode - Result in inches
81	0x51	Ranging Mode - Result in centimeters
82	0x52	Ranging Mode - Result in micro-seconds
83	0x53	ANN Mode - Result in inches
84	0x54	ANN Mode - Result in centimeters
85	0x55	ANN Mode - Result in micro-seconds
160	0xA0	1st in sequence to change I2C address
165	0xA5	3rd in sequence to change I2C address
170	0xAA	2nd in sequence to change I2C address

Ranging Mode

To initiate a ranging, one of the above commands must be written to the command register. The first echo range measured is placed in locations 2, 3; the second in 4, 5; etc. If a location (high and low bytes) is 0, then there will be no further reading in the rest of the registers. The default time for completion of ranging is 65ms; however it can be shorten by writing to the range register before issuing a ranging command. The SRF08 will not respond to any I^2C activity whilst ranging. Therefore, if it is tried to read from the sonar whilst ranging, 255 (0xFF) will return. As soon as the ranging is complete the SRF08 will respond again.

Changing the Range

The maximum range of the SRF08 is set by an internal timer. By default it is 65ms (equivalent to 11m of range). This is much further than the 6m that this sonar is actually capable of. It is possible to reduce the time the SRF08 listens for an echo, and hence the range, by writing to the range register at location 2. The range can be set in steps of about

43mm (0.043m or 1.68 inches) up to 11 meters. If it is necessary to reduce the range to be able to fire the SRF08 at a faster rate, it will be necessary to reduce the gain. The range is set to maximum every time the SRF08 is powered-up; then, if a different range is necessary, it must be changed as part of the system initialization code.

Analogue Gain

The analogue gain register sets the maximum gain of the analogue stages. To set it, the values of Table 9.3 must be written to the gain register at location 1.

Table A.3: Analogue Gain Values

Gain Register		Maximum Analogue Gain	Gain Register		Maximum Analogue Gain
Decimal Hex	Decimal Hex		Decimal Hex	Decimal Hex	
0	0x00	94	16	0x10	177
1	0x01	97	17	0x11	187
2	0x02	100	18	0x12	199
3	0x03	103	19	0x13	212
4	0x04	107	20	0x14	227
5	0x05	110	21	0x15	245
6	0x06	114	22	0x16	265
7	0x07	118	23	0x17	288
8	0x08	123	24	0x18	317
9	0x09	128	25	0x19	352
10	0x0A	133	26	0x1A	395
11	0x0B	139	27	0x1B	450
12	0x0C	145	28	0x1C	524
13	0x0D	152	29	0x1D	626
14	0x0E	159	30	0x1E	777
15	0x0F	168	31	0x1F	1025

The purpose of providing a limit to the maximum gain is to allow you to fire the sonar more rapidly than 65ms. Since the ranging can be very short, a new ranging can be initiated as soon as the previous range data has been read. To reduce cross talk, the maximum gain can be reduced to limit the modules sensitivity to the weaker distant echo, whilst still able to detect close by objects. The relationship between the gain register setting and the actual gain is not linear; the appropriated gain depends on the size, shape and material of the objects to be detected. The range and gain registers are automatically set by the SRF08 to their default values when it is powered-up.

Changing the I²C Bus Address

To change the I²C address of the SRF08, only one module may be attached to the bus. The sequence of 3 commands shown in Table 9.2 must be written in the correct order to the register at location 0, followed by the new address. The red LED on the module is used to flash out a code for the I²C address on power-up (see Table A.4). The flashing is terminated immediately on sending a command the SRF08.

Table A.4: I²C Addresses

Address		Long Flash	Short flashes	Address		Long Flash	Short flashes
Decimal	Hex			Decimal	Hex		
224	E0	1	0	240	F0	1	8
226	E2	1	1	242	F2	1	9
228	E4	1	2	244	F4	1	10
230	E6	1	3	246	F6	1	11
232	E8	1	4	248	F8	1	12
234	EA	1	5	250	FA	1	13
236	EC	1	6	252	FC	1	14
238	EE	1	7	254	FE	1	15

Power Consumption

The average current consumption measured is around 12mA during ranging, and 3mA standby. The module goes automatically to standby mode after a ranging, whilst waiting for a new command on the I²C bus. Table A.5 shows the current profile given by the manufacturer.

Table A.5: Power demand

Operation	*Current*	*Duration*
Ranging command received - Power on	275mA	3µs
+/- 10V generator Stabilization	25mA	600µs
8 cycles of 40kHz of sending	40mA	200µs
Ranging	11mA	65ms max
Standby	3mA	Indefinite

A.2 I2C Bus

Total bus capacitance needs to be less than 400 pF (e.g., about 20-30 devices or 10 m of trace) to respect rise and fall time requirements. Each device must be able to drive up to 3 mA for logic low level of 0.4mA on an open drain bus with pull-ups in the range from 2kOhms to 10kOhms. Bi-directional I²C bus buffers are available that isolate the capacitance on different legs of the bus, allowing larger (e.g., 2000pF) and longer (e.g., 2000m) buses.

Typically, the four most significant bits are fixed and assigned to specific categories of devices (e.g. 1010 is assigned to serial EEPROMs). The three less significant bits (e.g., A2, A1 and A0) are programmable through hardware address pins allowing up to eight different I²C address combinations and therefore allowing up to eight of that type of device to operate on the same I²C-bus. These pins are held high to VCC (1) or held low to GND (0).

Multi-master is the ability for more than one master to co-exist on the bus at the same time without collision or data loss. Typically bit-banged software implemented masters are not multi-master capable. Parallel to I²C bus controllers provide an easy way to add a multi-master hardware I²C port to DSPs and ASICs.

- Arbitration: it is the prearranged procedure, which authorizes only one master at a time to take control of the bus.
- Synchronization: it is the prearranged procedure, which synchronizes the clock signals provided by two or more masters.

I²C address of the targeted device is sent in the first byte and the least significant bit of this initial byte indicates if the master is going to send (write) or receive (read) data from the receiver, called the slave device. Each transmission sequence must begin with the *start* condition and end with the *stop* or *restart* condition. If there are two masters on the same I²C-

bus, there is an arbitration procedure if both try to take control of the bus at the same time by generating the *start* command at the same time. Once a master (e.g., microcontroller) has control of the bus, no other master can take control until the first master sends a *stop* condition and places the bus in an idle state.

Terminology for Bus Transfer

- *FREE* - the bus is free or idle; the data line SDA and the SCL clock are both in the high state.
- *START* or *RESTART* - data transfer begins with a Start condition. The level of the SDA data line changes from high to low, while the SCL clock line remains high. When this occurs, the bus becomes 'busy'.
- *CHANGE* - while the SCL clock line is low, the data bit to be transferred can be applied to the SDA data line by a transmitter. During this time, SDA may change its state, as long as the SCL line remains low.
- *DATA* - a high or low bit of information on the SDA data line is valid during the high level of the SCL clock line. This level must be kept stable during the entire time that the clock remains high to avoid misinterpretation as a Start or Stop condition.
- *STOP* - data transfer is terminated by a Stop condition. This occurs when the level on the SDA data line passes from the low state to the high state, while the SCL clock line remains high. When the data transfer has been terminated, the bus is free once again.

Bus Extension

As new I²C devices are operating at various voltage levels, different manufacturers has developed fully bi-directional data transfer circuit for I²C-devices operating from different supply voltages for almost no additional design-in effort or cost. The addition of two low-cost transistors, placed between the different voltage level sections of the I²C-bus, separates and transfers the logic voltage levels of the bus lines, as shown in Fig. A.1.

These transistors work as Bus-buffers and enable to expand the limitations of the I^2C-bus in many senses, but here the interest relies on the use longer cables. Since in general the I^2C-networks are internal to devices, it does not support long networks because of its capacitance limitation. If the network becomes too long, the overall capacitance becomes too high and the necessary time requirements for the logical level shifting cannot be achieved. As on ALDURO the sensors will be connected to the microcontroller in a by I^2C-network longer than 10m, the use of buffers is necessary, to guarantee the normal work of the bus. Moreover, as these buffers make the network to operate at a higher tension, an improvement in the noise rejection of the network occurs.

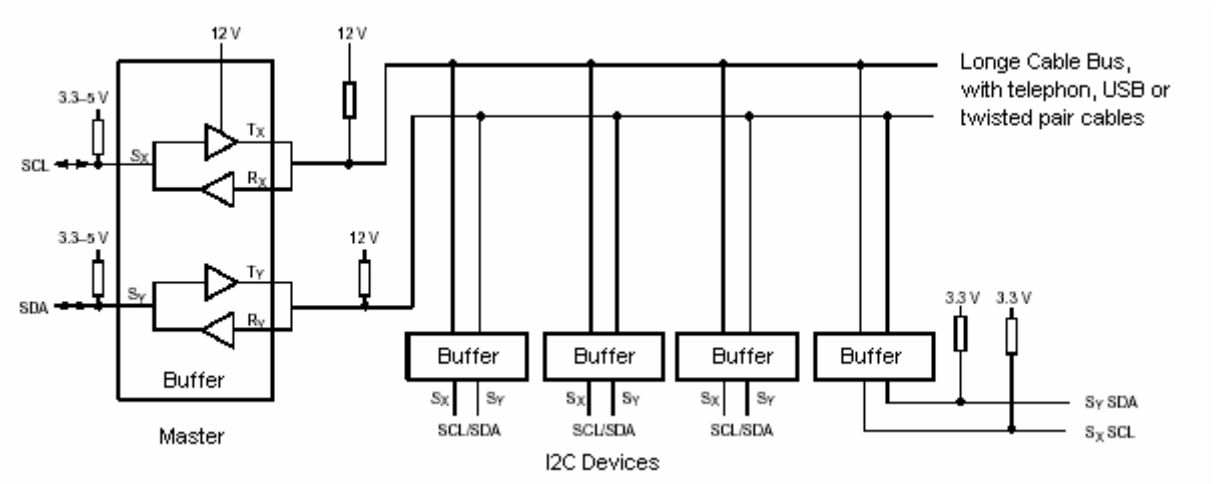

Fig. A.1: Extended I^2C-Bus

B Recursive Least-Squares Method for System Identification

In the general least-squares problem, the output of a linear model y is given by the linearly parameterized expression

$$y = \theta_1 \cdot f_1(\mathbf{u}) + \theta_2 \cdot f_2(\mathbf{u}) + \ldots + \theta_3 \cdot f_3(\mathbf{u}) \tag{B.1}$$

where $\mathbf{u} = [u_1, \ldots, u_p]^T$ is the input vector of the model; $f_1, \ldots, f_n$ are known functions of $\mathbf{u}$, and $\theta_1, \ldots, \theta_n$ are unknown parameters to be estimated. In statistics, the task of fitting data using a linear model is referred to as linear regression. Thus, Eq. B.1 is also called the regression function, and θ_i are called the regression coefficients. To identify the unknown parameters θ_i, it is usually necessary to perform experiments to obtain a training data set composed of data pairs $\{ (u_i; y_i), i = 1, \ldots, m\}$; they represent desired input-output pairs of the target system to be modeled. Substituting each data pair into Eq. B.1 yields a set of m linear equations:

$$\begin{cases} f_1(\mathbf{u}_1) \cdot \theta_1 \cdot + f_2(\mathbf{u}_1) \cdot \theta_2 + \ldots + f_3(\mathbf{u}_1) \cdot \theta_3 = y_1 \\ f_1(\mathbf{u}_2) \cdot \theta_1 \cdot + f_2(\mathbf{u}_2) \cdot \theta_2 + \ldots + f_3(\mathbf{u}_2) \cdot \theta_3 = y_2 \\ f_1(\mathbf{u}_3) \cdot \theta_1 \cdot + f_2(\mathbf{u}_3) \cdot \theta_2 + \ldots + f_3(\mathbf{u}_3) \cdot \theta_3 = y_3 \end{cases} \tag{B.2}$$

Using matrix notation, the preceding equations can rewritten in a concise form:

$$\mathbf{A} \cdot \boldsymbol{\theta} = \mathbf{y} \tag{B.3}$$

where **A** is an $m \times n$ matrix (sometimes called the design matrix):

$$\mathbf{A} = \begin{bmatrix} f_1(\mathbf{u}_1) & \cdots & f_n(\mathbf{u}_1) \\ \vdots & \ddots & \vdots \\ f_1(\mathbf{u}_m) & \cdots & f_n(\mathbf{u}_m) \end{bmatrix} \tag{B.4}$$

$\boldsymbol{\theta}$ is an $n \times 1$ unknown parameter vector and **y** is an $m \times 1$ output vector:

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_1 & \cdots & \theta_n \end{bmatrix}^T \tag{B.5}$$

$$\mathbf{y} = \begin{bmatrix} y_1 & \cdots & y_n \end{bmatrix}^T \tag{B.6}$$

The i^{th} row of the joint data matrix $[\mathbf{A} \vdots \mathbf{y}]$ denoted by $[\mathbf{a}_i^T \vdots y_i]$, is related to the i^{th} input-output data pair ($\mathbf{u}_i$; y_i) through

$$\mathbf{a}_i^T = [f_1(\mathbf{u}_i) \quad \cdots \quad f_n(\mathbf{u}i_1)] \tag{B.7}$$

Since most of our calculation is based on matrices **A** and **y**, sometimes is referred to $(\mathbf{a}_i^T; y_i)$ as the i^{th} data pair of the training data set. To identify uniquely the unknown vector $\boldsymbol{\theta}$, it is necessary that $m > n$. If **A** is square ($m = n$) and nonsingular, then the system in Eq. B.2 can be solved as:

$$\boldsymbol{\theta} = \mathbf{A}^{-1} \cdot \mathbf{y} \tag{B.8}$$

However, usually m is greater than n, indicating that there are more data pairs than fitting parameters. In this case, an exact solution satisfying all the m equations is not always possible, since the data might be contaminated by noise, or the model might not be appropriate for describing the target system. Thus, Eq. B.3 should be modified by incorporating an error vector **e** to account for random noise modeling error, as follows:

$$\mathbf{A} \cdot \boldsymbol{\theta} + \mathbf{e} = \mathbf{y} \tag{B.9}$$

Now, instead of finding the exact solution to Eq. B.3, a $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ has to be found, which minimizes the sum of squared error defined by

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{m} \left(y_i - \mathbf{a}_i^T \cdot \boldsymbol{\theta}\right)^2 = \mathbf{e}^T \cdot \mathbf{e} = \left(\mathbf{y} - \mathbf{A} \cdot \boldsymbol{\theta}\right)^T \cdot \left(\mathbf{y} - \mathbf{A} \cdot \boldsymbol{\theta}\right) \tag{B.10}$$

where $\mathbf{e} = \mathbf{y} - \mathbf{A}\boldsymbol{\theta}$ is the error vector produced by specific choice of $\boldsymbol{\theta}$. As $E(\boldsymbol{\theta})$ is in quadratic form, it has a unique minimum at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$. Thus, calculating the derivative of $E(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$:

$$\frac{\partial E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 2 \cdot \mathbf{A}^T \cdot \mathbf{A} \cdot \boldsymbol{\theta} - 2 \cdot \mathbf{A}^T \cdot \mathbf{y} \tag{B.11}$$

By setting the derivative of the sum of the squared equal to 0, it is obtained the so called *normal equation* and the optimal vector of parameters:

$$\mathbf{A}^T \cdot \mathbf{A} \cdot \hat{\boldsymbol{\theta}} = \mathbf{A}^T \cdot \mathbf{y} \tag{B.12}$$

$$\hat{\boldsymbol{\theta}} = \left(\mathbf{A}^T \cdot \mathbf{A}\right)^{-1} \cdot \mathbf{A}^T \cdot \mathbf{y} \tag{B.13}$$

The derived least-squares estimator can be expressed as:

$$\boldsymbol{\theta}_k = \left(\mathbf{A}^T \cdot \mathbf{A}\right)^{-1} \cdot \mathbf{A}^T \cdot \mathbf{y} \tag{B.14}$$

where (^) is left out the hat for simplicity. Here it is assumed that the row dimensions for **A** and **y** are *k*; thus a subscript *k* is added in the preceding equation to denote the number of data pairs used for the estimator $\boldsymbol{\theta}$. The *k* also can be looked as measure of time if the data pairs become available in sequential order. In this way, a new data pair ($\mathbf{a}^T$; *y*) becomes available as the entry of order *m* + l in the data set. Then, instead of using all the *k* + l available data pairs to recalculate the Lest Squares Estimator $\boldsymbol{\theta}_{k+1}$, it would be interesting to take advantage of the $\boldsymbol{\theta}_k$ already available to obtain $\boldsymbol{\theta}_{k+1}$ with a minimum of effort. That means to use the new data pair ($\mathbf{a}^T$; *y*) to update $\boldsymbol{\theta}_k$. This problem is called recursive least-squares identification. Obviously, $\boldsymbol{\theta}_{k+1}$ can be expressed as:

$$\hat{\boldsymbol{\theta}}_{k+1} = \left(\begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix} \right)^{-1} \cdot \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{y} \\ y \end{bmatrix} \tag{B.15}$$

To simplify the notation, two $n \times 1$ matrices $\mathbf{P}_k$ and $\mathbf{P}_{k+1}$ are introduced, which are defined as:

$$\mathbf{P}_k = \left(\mathbf{A}^T \cdot \mathbf{A}\right)^{-1} \tag{B.16}$$

$$\mathbf{P}_{k+1} = \left(\begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{A} \\ \mathbf{a}^T \end{bmatrix} \right)^{-1} = \left(\mathbf{A}^T \mathbf{A} + \mathbf{a}\mathbf{a}^T\right)^{-1} \tag{B.17}$$

These two matrices are related by:

$$\mathbf{P}_k^{-1} = \mathbf{P}_{k+1}^{-1} - \mathbf{a}\mathbf{a}^T \tag{B.18}$$

Using $\mathbf{P}_k$ and $\mathbf{P}_{k+1}$, is obtained:

$$\begin{cases} \boldsymbol{\theta}_k = \mathbf{P}_k \cdot \mathbf{A}^T \cdot \mathbf{y} \\ \boldsymbol{\theta}_{k+1} = \mathbf{P}_{k+1} \cdot \left(\mathbf{A}^T \cdot \mathbf{y} + \mathbf{a} \cdot y \right) \end{cases} \tag{B.19}$$

To express $\boldsymbol{\theta}_{k+1}$ in terms of $\boldsymbol{\theta}_k$, $\mathbf{A}^T\mathbf{y}$ must be eliminated from Eq. B.19. Then, from the first equation in Eq. B.19:

$$\mathbf{A}^T \cdot \mathbf{y} = \mathbf{P}_k^{-1} \cdot \boldsymbol{\theta}_k \tag{B.20}$$

By plugging this expression into the second in Eq. B.19 and applying Eq. B.18:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{P}_{k+1} \cdot \mathbf{a} \cdot \left(y - \mathbf{a}^T \cdot \boldsymbol{\theta}_k \right) \tag{B.21}$$

Thus, $\boldsymbol{\theta}_{k+1}$ can be expressed as a function of the old estimate $\boldsymbol{\theta}_k$ and the new data pairs ($\mathbf{a}^T$; y). It is interesting to note that Eq. B.21 has an intuitive interpretation: the new estimator $\boldsymbol{\theta}_{k+1}$ is equal to the old estimator $\boldsymbol{\theta}_k$ plus a correcting term based on the new data ($\mathbf{a}^T$; y). This correcting term is equal to an adaptive gain vector $\mathbf{P}_{k+1}\mathbf{a}$ multiplied by the prediction error produced by the old estimator, that is, y-$\mathbf{a}^T\boldsymbol{\theta}_k$.

Calculating $\mathbf{P}_{k+1}$ by Eq. B.17 involves the inversion of an $n \times n$ matrix. This is computationally expensive; therefore, an incremental formula for $\mathbf{P}_{k+1}$ has to be found. From Eq. B.18:

$$\begin{aligned} \mathbf{P}_{k+1} &= \left(\mathbf{P}_k^{-1} - \mathbf{a} \cdot \mathbf{a}^T \right)^{-1} \\ &= \mathbf{P}_k - \frac{\mathbf{P}_k \cdot \mathbf{a} \cdot \mathbf{a}^T \cdot \mathbf{P}_k}{1 + \mathbf{a}^T \cdot \mathbf{P}_k \cdot \mathbf{a}} \end{aligned} \tag{B.22}$$

In summary, the recursive least-squares estimator for the problem $\mathbf{A}\boldsymbol{\theta} = \mathbf{y}$, where the row of order k ($1 \le k \le m$) of $[\mathbf{A} \vdots \mathbf{y}]$ denoted by $[\mathbf{a}_i^T \vdots y_i]$, is sequentially obtained and can be calculated as follows:

$$\begin{aligned} \mathbf{P}_{k+1} &= \mathbf{P}_k - \frac{\mathbf{P}_k \cdot \mathbf{a}_{k+1} \cdot \mathbf{a}_{k+1}^T \cdot \mathbf{P}_k}{1 + \mathbf{a}_{k+1}^T \cdot \mathbf{P}_k \cdot \mathbf{a}_{k+1}} \\ \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \mathbf{P}_{k+1} \cdot \mathbf{a}_{k+1} \cdot \left(y_{k+1} - \mathbf{a}_{k+1}^T \cdot \boldsymbol{\theta}_k \right) \end{aligned} \tag{B.23}$$

Remarks:

- The matrix $\mathbf{P}_k$ is proportional to the covariance of the estimators.
- The least-squares estimator can be interpreted as a Kalman filter, where k is a time index, $\mathbf{e}(k)$ is random noise, $\boldsymbol{\theta}(k)$ is the state to be estimated, and $y(k)$ is the observed output.

- When extra parameters are introduced for better performance, $\boldsymbol{\theta}$ will have more components and there will be additional columns in matrix **A**. It is possible to reduce the complexity of the calculation by employing recursion in the number of parameters.

- From Eq. B.23 it is possible to see that the system would have a fast convergence if **P** is initially very large and $\boldsymbol{\theta}$ is very small. Therefore, to start the calculation, **P** is set as a diagonal matrix with very large values, and $\boldsymbol{\theta}$ is set to null.

References

[1] Akbarally, H. and Kleeman, L.: 3D Robot Sensing from Sonar and Vision. Proceedings of the 1996 IEEE International Conference on Robotics and Automation (1996), pp.686-691.

[2] Akbarally, H. and Kleeman, L.: A Sonar Sensor for Accurate 3D Target Localisation and Classification. IEEE International Conference on Robotics and Automation. Nagoya, Japan, May 1995, pp. 3003-3008995.

[3] Arakawa, K. and Krotkov, E.: Estimating Fractal Dimensions of Natural Terrain from Irregularly Spaced Data. Proceedings 1993IEEE/RSJ International Conference on Intelligent Rabats and Systems (1993), pp. 1364-1370.

[4] Arkin, R.C.: Reactive Robotic Systems. In: Handbook of Brain Theory and Neural Networks, ed. M. Arbib. 1995. MIT Press, pp. 793-796.

[5] Armstrong II, D.: Position and Obstacle Detection Systems for an Outdoor, Land-Based, Autonomous Vehicle. Master's Thesis, University of Florida, Gainesville, Florida (1993).

[6] Asada, M.: Map Building for a Mobile Robot from Sensory Data. IEEE Transactions on Systems, Man, and Cybernetics, 37-6 (1990), pp.1326-1336.

[7] Asada, M., Fukui, Y. and Tsuji, S.: Representing Global World of a Mobile Robot with Relational Local Maps. IEEE Transactions on Systems, Man and Cybernetics, 20-6(1990), pp. 1456-1461.

[8] Atrash, A. and Koenig, S.: Probabilistic Planning for Behavior-Based Robots. Proceedings of the International FLAIRS conference (FLAIRS), 2001, pp. 531-535.

[9] Banta, J. and Abidi, M.: Autonomous Placement of a Range Sensor for Acquisition of Optimal 3D Models. IEEE IECON, March (1996), pp. 1583-1588.

[10] Banta, J. and Abidi, M.: Optimal Range Sensor Position for 3D Model Reconstruction. Proceedings of the World Automation Congress, May (1996), pp. 57-62.

[11] Barshan, B. and Kuc, R.: Differentiating sonar reflections from corners and planes employing an intelligent sensor. IEEE Trans. on Pattern Analysis and Machine Intelligence, 12(6) (1990), pp. 560-569.

[12] Borenstein, J. and Koren, Y.: Real-time Obstacle avoidance for Fast Mobile Robots. In IEEE Transactions on Systems, Man, and Cybernetics. Sept. /Oct. 1989. Vol. 19, No.5, pp. 1179-1187.

[13] Borenstein, J. and Koren, Y.: Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance IEEE Journal of Robotics and Automation, 7th April (1991), pp. 535-539.

[14] Bozma, O. and Kuc, R.: Building a sonar map in a specular environment using a single mobile sensor. IEEE Trans. on Pattern Analysis and Machine Intelligence, 13(12) (1991), pp. 1260-1269.

[15] Braunstingl, R. and Ollero, A.: Evaluating the Wall Following Behavior of a Mobile Robot with Fuzzy Logic. IFAC/IMACS International Workshop on artificial Intelligence in Real Time Control. Bled, Slovenia, Nov (1995), pp. 89-93.

[16] Braunstingl, R., Mujika, J. and Uribe, J.P.: A Wall Following Robot with a Fuzzy Logic Controller Optimized by a Genetic Algorithm. FUZZ-IEEE/IFES'95 Fuzzy Robot Competition. Yokohama, Japan, March (1995), Vol.V, pp. 77-82.

[17] Braunstingl, R., Sanz, P. and Ezkerra, J.M.: Fuzzy Logic Wall Following of a Mobile Robot Based on the Concept of General Perception. ICAR'95, 7th International Conference on Advanced Robotics. Sep., 1995.Saint Feliu de Guixols, Spain. Vol. 1, pp. 367-376.

[18] Braunstingl, R.: Improving Reactive Navigation Using Perception Tracking. In: Jamshidi, M., Pin, F. and Dauchez, P., Robotics and Manufacturing, Recent Trends in Research and Applications. New York, 1996.Vol. 6, pp. 25-30.

[19] Burgard, W., Fox, D. and Henning, D.: Fast grid-based position tracking for mobile robots. KI - Kunstliche Intelligenz, 1997, pp. 289-300.

[20] Burgard, W., Fox, D., Jans, H., Matenar, C. and Thrun, S.: Sonar-based mapping with mobile robots using EM. Proc. 16th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1999, pp. 67-76.

[21] Castellanos, J., Neira, J. and Tardos, J.: Multisensor Fusion for Simultaneous Localization and Map Building. IEEE Transactions on Robotics and Automation, 17th Jun (2001), pp. 908-914.

[22] Caurin, G. A. de P.: Control of Walking Robots on Natural Terrain. Prof. G. Schweitzer, Prof. M. Hiller. ETH-Dissertation 10898, 1994.

[23] Chong, K.S. and Kleeman, L.: Feature-based Mapping in Real, Large Scale Environments using an Ultrasonic Array. In: International Journal Robotics Research. 1999. Vol. 18, nº1, pp. 3-19.

[24] Choset, H. and Burdick, J.W.: Sensor Based Planning for a Planar Rod Robot. In: Proceedings of the IEEE Int. Conf. on Robotics and Automation, 1996.

[25] Chya, A., Nagashima, Y. and Yuta, S.: Exploring Unknown Environment and Map Construction Using Ultrasonic Sensing of Normal Direction of Walls. Proceedings IEEE International Conference on Robotics and Automation (1994), pp. 485-492.

[26] DeBerg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O.: Computational Geometry Algorithms and Applications, Springer, Berlin (2000).

[27] Delgado, M., Gomez, A., Martinez, H. and Garcia, P.: Fuzzy Range Sensor Filtering for Reactive Autonomous Robots. 5th International Conference on Soft Computing. Oct 1998. Iizuka, JAPAN.

[28] Diaz, J.F., Stoytchev, A. and Arkin, R.C.: Exploring Unknown Structured Environments.14th International Fairs Conference, Key West, FL, May 2001.

[29] Driankov, D., Hellendorn, H. and Reinfrank, M.: An Introduction to Fuzzy Control. Springer Verlag, Berlin, 1996.

[30] Dudek, G., Freedman, P. and Rekleitis, I. M.: Just-in-time sensing: efficiently combining sonar and laser range data for exploring unknown worlds. IEEE International Conference on Robotics and Automation, Minneapolis, (1996), pp. 667-672.

[31] Elfes, A.: Using occupancy grids for mobile robot perception and navigation. IEEE Computer Magazine, special issue on Autonomous Intelligent Machines, (1989). Vol. 22 (6), pp. 46-58.

[32] Elfes, A.: Occupancy Grids: A probabilistic framework for robot perception and navigation. PhD dissertation, Carnegie Mellon University, Department of Electrical and Computer Engineering, (1989).

[33] Faceli, K., de Carvalho, A.C.P.L.F. and Rezende, S.O.: Experimentos em aprendizado de Máquina para Fusão de Sensores. Proceedings of the V Brazilian Conference on Neural Networks. April 2-5, 2001. Rio de Janeiro - Brazil, pp.301-306.

[34] Fox, D., Burgard, W. and Thrun, S.: A Hybrid Collision Avoidance Method for Mobile Robots. Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium, 1998.

[35] Frank, A.A.: Automatic Control Systems for Legged Locomotion. PhD Thesis, University of Southern California, Los Angeles, 1968.

[36] Gambino, F., Oriolo, G. and Ulivi, G.: A comparison of three uncertainty calculus techniques for ultrasonic map building. Proceedings of the 1996 SPIE International Symposium on Aerospace / Defense Sensing and Control, 1996, Orlando, USA, pp. 249-260.

[37] Garcia, M.: Reconstruction of Visual Surfaces from Sparse Data Using Parametric Triangular Approximations. IEEE International Conference on Image Processing, 2(1994), pp. 750-754.

[38] Gourley, C. and Trived, M.: Sensor Based Obstacle Avoidance and Mapping for Fast Mobile Robots. Proceedings IEEE International Conference on Robotics and Automation, 2(1994), pp. 1306-1311.

[39] Gowdy, J., Stentz, A. and Herbert, M.: Hierarchical Terrain Representations for Off-road Navigation. Proceedings of SPIE - Mobile Robots Vol. 1388 (1990), pp:131-140.

[40] Graft, L. and Usery, E.: Automated Classification of Generic Terrain Features in Digital Elevation Models. Photogram metric Engineering and Remote Sensing, (1993), Vol. 59-9, pp. 1409-1417.

[41] Gruyer, D. and Berge-Cherfaoui, V.: Increasing sensor data reliability by using of a Fuzzy Estimator-Predictor. AVACS ‘98 - International Conference on advances in Vehicle Control and Safety. Amiens, France, July 1998.

[42] Haddad, H., Khatib, M., Lacroix, S. and Chatila, R.: Reactive Navigation in Outdoor Environments using Potential Fields. Proceedings of the 1998 IEEE International Conference on Robotics and Automation (1998), pp. 1232-1237.

[43] Hager, G. and Mintz, M.: Computational Methods for Task-directed Sensor Data Fusion and Sensor Planning. The International Journal of Rabattes Research, 104(1991), pp. 285-313.

[44] Hancock, J., Hebert, M. and Thorpe, C.: Laser Intensity-Based Obstacle Detection. Proceedings of the 1998IEEE/RSJInternational Conference an Intelligent Robots and Systems, (1998), pp. 1541-1546.

[45] Heale, A. and Kleeman, L.: A Real Time DSP Sonar Echo Processor. IEEE/RSJ International Conference on Intelligent Robots and Systems. Takamatsu, Japan, Oct 2000, pp. 1261-1266.

[46] Hiller, M. Germann, D. and Müller, J.: Free gait pattern generator for the quadruped-walking robot ALDURO. Proceedings of the 10th International Symposium on Dynamic Problems of Mechanics X-DINAME, Edited by Kurka, P.R.G and Fleury, A.T., Ubatuba, Brazil, 10-14 March 2003.

[47] Hiller, M. and Müller, J.: Design und Realisierung des autonomen Schreitfahrwerks ALDURO. Proceedings der VDI-EKV Tagung Mechatronik - Mechanisch/Elektrische Antriebstechnik, Wiesloch, 2000.

[48] Hirose, S., Tsukagoshi, H. and Arikawa, K.: Development of Quadruped Walking Robots, TITAN-VII and TITAN VIII, Final Report, Research Program on Mechanism for Emergent Machine Intelligence, Japan, (1999), pp.535-545.

[49] Hogan, J.E., Schmitt, P.R. and Book, W.J.: Ultrasonic Sensor Model and Configuration Simulations for Mobile Robot Navigation in Narrow Aisles, Westinghouse Savannah River Company through ERDA December 9, 1994.

[50] Hoover, A. and Olsen, B.: A Real-Time Occupancy Map from Multiple Video Streams. Proceedings of the 1999 IEEE International Conference on Robotics and Automation (1999), pp. 2261-2266.

[51] Howard, A. and Kitchen, L.: Generating Sonar Maps in Highly Specular Environments. Proceedings of the Fourth International Conference on Control, Automation, Robotics, and Vision. (1996), pp. 1870-1874.

[52] Howard, A., Tunstel, E., Edwards, D., and Carlson, A. Enhancing Fuzzy Robot Navigations Systems by Mimicking Human Visual Perception of Natural Terrain Traversability. Joint 9th IFSA World Congress and 20th NAFIPS International Conference. Vancouver, Canada, (2001), pp. 7-12.

[53] Jang, J., Sun, C. and Mizutani, E.: Neuro-Fuzzy and Soft Computing. Prentice Hall, Upper Saddle River-USA, 1997.

[54] Jörg, K.: World Modeling for an Autonomous Mobile Robot Using Heterogeneous Sensor Information. Robotics and Autonomous Systems, 14(1995), pp. 159-170.

[55] Kecskeméthy, A.: A spatial leg mechanism with anthropomorphic properties for ambulatory robots. In Lenarcic, J. and Ravani, B. (editors); Advances in Robot Kinematics and Computational Geometry, Kluver Academic Publishers, Dordrecht, Boston, London, 1994, pp. 161-170.

[56] Kelly, A. and Stentz, A.: Rough Terrain Autonomous Mobility - Part 2: An Active Vision, Predictive Control Approach. Autonomous Robots, 5(1998):163-198.

[57] Kelly, A., Stentz, A. and Herbert, M.: Terrain Map Building for Fast Navigation on Rugged Outdoors Terrain. Proceedings of SPIE - Mobile Robots VII, 1831(1992), pp. 576-589.

[58] Kleeman, L. and Kuc, R.: Mobile Robot Sonar for Target Localization and Classification. In: International Journal of Robotics Research. August 1995. Vol. 14, nº 4, pp. 295-318.

[59] Kleeman, L.: Advanced Sonar Sensing. Proceedings 10th International Symposium Robotics Research. Lorne Victoria, Australia. November 2001, pp. 286-295.

[60] Kleeman, L.: Advanced Sonar with Velocity Compensation. International Journal Robotics Research, Vol. 23 Nr. 2, Feb 2004, pp 111-126.

[61] Klein, L.: A Boolean Algebraic Approach to Multiple Sensors Voting Fusion. IEEE Transactions on Aerospace and Electronic Systems. 29-2(1993), pp. 317-327.

[62] Koenig, S. and Likhachev, M.: Improved Fast Replanning for Robot Navigation in Unknown Terrain. Technical Report GITCOGSCI-2002/3, College of Computing, Georgia Institute of Technology, Atlanta, Georgia, 2001.

[63] Krotkov, E. and Hoffman, R.: Terrain Mapping for a Walking Planetary Rover. IEEE Transactions on Robotics and Automation, 10-6 (1994), pp. 728-739.

[64] Krotkov, E. and Herbert, M.: Mapping and Positioning for a Prototype Lunar Rover. IEEE International Conference on Robotics and Automation (1995), pp. 2913-2919

[65] Kuc, R. and Siegel, M.W.: Physically based simulation model for acoustic sensor robot navigation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 9(6) (1987), pp. 766-778.

[66] Kweon, I. and Kanade, T.: Extracting Topography Features for Outdoors Mobile Robots. Proceedings oft he 1991 IEEE International Conference on Robotics and Automation (1991), pp.1992-1997.

[67] Kweon, I. and Kanade, T.: High-Resolution Terrain Map from Multiple Sensor Data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 14-2(1992), pp. 278-292.

[68] Lacroix, S., Chatila, R., Fleury, S., Herrb, M. and Simon, T.: Autonomous Navigation in Outdoor Environment Adaptive Approach and Experiment. IEEE Conference on Robotics and Automation, Vol. l (1994), pp. 426-432.

[69] Lamela, H., Lopez, J. and Garcia, E.: Low Cost Range map Obtaining for Mobile Robots Based on a Triangulation Rangefinder. Proceedings of the 1997 23rd Annual International Conference on Industrial Electronics, Control, and Instrumentation, IECON (1997), pp. 1284-1287.

[70] Langer, D. and Jochem, T.: Fusing Radar and Vision for Detecting, Classifying and Avoiding Roadway Obstacles. Proceedings of the 1996 IEEE Intelligent Vehicles Symposium (1996), pp. 333-338.

[71] Leonard, J., Durrant-Whyte, H. and Cox, I.: Dynamic Map Building for an Autonomous Mobile Robot. The International Journal of Robotics Research, ll-4 (1992), pp. 286-298.

[72] Leung, M. and Huang, T.: An Integrated Approach to 3-D Motion Analysis and Object Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13-10(1991), pp. 1075-1084.

[73] Leyden, M., Toal, D. and Flanagan, C.: A Fuzzy Logic Based Navigation System for a Mobile Robot. In: Automatisierungssymposium, Wismar, 1999.

[74] Li, F.: Modeling the Ground Plane Transformations for Real-Time Obstacle Detection. Computer Vision and Image Understanding, 71-1(1998), pp.137- 152.

[75] Liao, X., Cao, M., Cao, J. and Hall, E.L.: A Probabilistic Model for AGV Mobile Robot Ultrasonic sensor. In: Proceedings of the SPIE Intelligent Robots and Computer Vision Conference XVIII. 1999. Sep. 19-22. Boston, MA.

[76] Lim, J. and Cho, D.: Experimental Investigation of Mapping and Navigation Based on Certainty Grids Using Sonar Sensors. Robotica, 11-1(1993), pp.7-17.

[77] Little, C. and Wilson, C.: Rapid World Modeling for Robotics. Proceedings of the World Automation Congress (1996):441-446.

[78] Maio, D. and Rize, S.: Map Learning and Clustering in Autonomous Systems. IEEE Transactions on Pattern Analysis and Machine Intelligence, 15-21(1993), pp. 1286-1297.

[79] Matthies, L., Kelly, A., Litwin, T. and Tharp, G.: Obstacle Detection for Unmanned Ground Vehicles: A Progress Report. International Symposium of Robotics Research (1995), pp. 66-71.

[80] Mäzl, R. and Pfeucil, L.: Building a 2D Environment Map from Laser Range-Finder Data. Proceedings of the IEEE Intelligent Vehicles Symposium (2000), pp. 290-295.

[81] McKeown Jr., D.: Toward Automatic Cartographic Feature Extraction. Mapping and Spatial Modeling for Navigation, 65(1990): 149-180.

[82] McKerrow, P.: Robot Perception with Ultrasonic Sensors Using Data Fusion. Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics, 2 (1995), pp. 1380- 1385.

[83] Moravec, H.P. and Elfes, A.: High Resolution Maps from Wide Angle Sonar. Proceedings of the 1985's IEEE International Conference on Robotics and Automation, 1985.

[84] Moravec, H.: Sensor Fusion in Certainty Grids for Mobile Robots. AI Magazine, Summer (1988), pp. 61-74.

[85] Morgado de Gois, J.A., Germann, D. and Hiller, M.: Sensor-based ground detection in unstructured terrain for the walking-machine ALDURO. Proceedings of the 6th International Conference on Climbing and Walking Robots CLAWAR, 2003, Catania, Italy, pp. 911-918.

[86] Morgado de Gois, J.A. and Hiller, M.: Ultrasound sensor system with fuzzy data processing. Proceedings of the 7th International Conference on Climbing and Walking Robots CLAWAR, 2004, Madrid, Spain.

[87] Morgado de Gois, J.A., Germann, D. and Hiller, M.: Design and Control of a Quadruped Robot Walking in Unstructured Terrain. IEEE Conference on Control Applications 2004, Taipei, Taiwan, September 2--4, 2004.

[88] Morgado de Gois, J.A. and Hiller, M.: Implementation of a Sensor System for Obstacle Avoidance at a Four-Legged Robot. Proceedings of the XI DINAME, Edited by D.A. Rade and V. Steffen Jr, Ouro Preto – MG, Brazil, 28th Feb - 4th March, 2005.

[89] Mosher, R.S.: Exploring the potential of a Quadruped. International Automotive Engineering Congress, Society of Automotive Engineers Inc, Detroit, Michigan, Jan. 1969.

[90] Müller, J.: Entwicklung einer virtueller Prototypen des hydraulisch angetriebenen Schreitfahrwerks ALDURO. Fortschr.-Ber. VDI Reihe 1 Nr.356. Düsseldorf: VDI Verlag 2002.

[91] Müller, J., Schneider, M. and Hiller, M.: Modeling and simulation of the large-scale hydraulically driven ALDURO. Proceedings of the EUROMECH Colloquium, Munich, 1998, p. 375 and Proceedings of the Workshop Autonomous Walking, Magdeburg, Germany.

[92] Murphy, R.: Dempster-Shafer Theory for Sensor Fusion in Autonomous Mobile Robots. IEEE Transactions on Robotics and Automation, (1998): 197-206.

[93] Nashashibi, F., Devy, M. and Fillatreau, P.: Indoor Scene Terrain Modeling Using Multiple Range Images for Autonomous Mobile Robots. Proceedings of the 1992 IEEE International Conference on Robotics and Automation, (1992), pp. 40-46.

[94] Ohya, A., Nagashima, Y. and Yuta, S.: Exploring Unknown Environment and Map Construction Using Ultrasonic Sensing of Normal Direction of Walls. Proceedings of the IEEE International Conference on Robotics and Automation, l (1994):485-492.

[95] Olin, K. and Hughes, D.: Autonomous Cross-country Navigation – An Integrated Perception and Planning System. IEEE Expert. August (1991): 16-30.

[96] Oriolo, G., Vendittelli M. and Ulivi, G.: On-Line Map Building and Navigation for Autonomous Mobile Robots. IEEE International Conference on Robotics and Automation (1995):2900-2906.

[97] Oriolo, G., Ulivi, G. and Venditelli, M.: Real-time map building and navigation for autonomous robots in unknown environments. IEEE Transactions on Systems, Man and Cybernetics, 1999, pp. 100-137.

[98] Pavlin, G. and Braunstingl, R.: Constrained World Modeling With Wide-Angle Rangefinders. 8th Int. Workshop on Robotics in Alpe-Adria-Danube Region. 1999. RAAD ‘99, München.

[99] Pavlin, G. and Braunstingl, R.: Context-Based Feature Extraction with Wide-Angle Sonars. Proceedings of International Conference on Intelligent Robots and Systems, Takamatsu, Japan, 2000.

[100] Plaza, M., Prieto, S., Davila, O., Poblaciön, O. and Luna, D.: An Obstacle Detector System Based on Laser Technology. Circuits and Devices, September (2001):9-15.

[101] Ram, A. and Santamaría, J. C.: A Multistrategy Case-Based and Reinforcement Learning Approach to Self-Improving Reactive Control systems for Autonomous Robotic Navigation. In Proceeding of the Second International Workshop on Multistrategy Learning. May 1993. Harpers Ferry, WV.

[102] Risse, W.: Sensorgestützte Bewegungssteuerung kinematisch redundanter Roboter. Bericht aus der Robotik, Aachen: Shaker Verlag, 2002.

[103] Samet, H.: The Design and Analysis of Spatial Data Structures, Addison-Wesley Publishing Company, Inc., Reading Massachusetts (1990).

[104] Schneider, F., Wolf, H. and Holzhausen, K.: Acquisition and Display of a World Model Using an Autonomous Mobile Land Robot. IEEE/RSJ/G1 International Conference on Intelligent Robots and Systems 2(1994), pp. 769-775.

[105] Shao, M., Chellappa, R. and Simehony, T.: Reconstruction a 3D Depth Map from One or More Images. CVGIP: Image Understanding, 53-2(1991), pp. 219-226.

[106] Slack, M.: Fixed Computation Real-Time Sonar Fusion for Local Navigation. Proceedings of the IEEE International Conference on Robotics and Automation, 2(1993), pp. 123-129.

[107] Smurb, R. and Everett, H.R.: Intelligent Sensor Fusion for a Mobile Security Robot. Sensors, March (1994), pp. 18-28.

[108] Stark, L. and Bowyer, K.: Achieving Generalized Object Recognition through Reasoning about Association of Function to Structure. IEEE Transactions on Pattern Analysis and Machine Intelligence. 13-10(1991), pp.1097-1103.

[109] Strat, T. and Fishler, M.: Context-Based Vision: Recognizing Objects Using Information from both 2-D and 3-D Imagery. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13-10(1991): 1050-1065.

[110] Stuck, E., Manz, A., Green, D. and Elgazzar, S.: Map Updating and Path Planning for Real-Time Mobile Robot Navigation. IEEE/RSJ/GI International Conference on Intelligent Robots and Systems. l(1994). pp. 431-438.

[111] Sugeno, M. and Kang, G.T.: Structure identification of Fuzzy model. Fuzzy sets and Systems, (1988) Vol.28, pp.15-33,

[112] Suorsa, R. and Sridhar, B.: A Parallel Implementation of a Multi-sensor Feature-Based Range-Estimation Method. IEEE Transactions of Robotics and Automation, 10-6 (1994), pp. 755-768.

[113] Sutherland, K., and Thompson, W.: Localizing in Unstructured Environments: Dealing with Errors. IEEE Transactions on Robotics and Automation, 10-6(1994), pp. 740-754.

[114] Silva, A., Menezes, P. and Dias, J.: Grid Based Navigation for Autonomous Robots - An Algorithm Based on the Integration of Vision and Sonar Data. Proceedings of the 1997 IEEE International Symposium on Industrial Electronics, ISIE, 3(1997), pp. 802-806.

[115] Takagi, T. and Sugeno, M.: Fuzzy identification of systems and applications to modeling and control. IEEE Transactions on Systems, Man and Cybernetics (1985), pp. 116-132.

[116] Tsourveloudis, N.C., Valavanis, K.P. and Hebert, T.: Autonomous Vehicle Navigation Utilizing Electrostatic Potential Fields and Fuzzy Logic. IEEE Transactions on Robotics and Automation. August, 2001. vol. 17, Nr. 4 pp. 490-497.

[117] Unmanned Ground Vehicles/Systems Joint Project Office, Joint Architecture for Unmanned Ground Systems (JAUGS): Volume I Domain Model, Alabama (2000).

[118] Unmanned Ground Vehicles/Systems Joint Project Office, Joint Architecture for Unmanned Ground Systems (JAUGS): Volume II Reference Architecture Specification, Alabama (2000)

[119] Van Dam, J.W.M., Kröse, B.J.A. and Groen, F.C.A.: Adaptive Sensor Models. Proceedings of 1996 International Conference on Multi-sensor Fusion and Integration for Intelligent Systems, Washington D.C., December 1996, pp. 705-712.

[120] Waxman, A., LeMoigne, J., Davis, L., Srinivason, B., Kushner, T., Liang, E. and Siddalingaiah, T.: A Visual Navigation System for Autonomous Land Vehicles. IEEE Journal of Robotics and Automation, 2 (1987), pp. 124-141.

[121] Wijk, O., Jensfelt, P. and Christensen, H.: Triangulation Based Fusion of Ultrasonic Sensor Data. Proceedings of the 1998 IEEE International Conference on Robotics and Automation (1998), pp. 3419-3242.

[122] Yata, T., Ohya, A. and Yuta, S.: Fusion of Omni-directional Sonar and Omni-directional Vision for Environment Recognition of Mobile Robots. Proceedings of the IEEE International Conference on Robotics and Automation, (2000), pp. 3925-3930.

[123] Yi, Z., Khing, Y., Seng, C. and Wei, Z.: Multi-Ultrasonic Sensor Fusion for Mobile Robots. Proceedings of IEEE Intelligent Vehicles Symposium (2000), pp. 387-391.

[124] Zadeh, L.: Fuzzy sets, Inf. Control 8, 1965, pp. 338-353.

[125] Zadeh, L.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 1, 1978, pp. 3-28.

[126] Zadeh, L.: Uncertainty in Knowledge Bases. Bouchon-Meunier, R.R. Yager and L.A. Zadeh (Eds.), Springer-Verlag, Berlin, 1991.

[127] Zadeh, L.: Probability Theory and Fuzzy Logic are Complementary rather than Competitive, Technometrics, 1995, Vol 37, pp. 271-276.

[128] Zhang, Z. and Fangeras, O.: A 3D World Model Builder with a Mobile Robot. The International Journal of Robotics Research, ll-4(1992), pp. 269-285.

[129] Zhu, Z., Xu, G., Chen, S. and Lin, X.: Dynamic Obstacle Detection through Cooperation of Purposive Visual Modules of Color, Stereo, and Motion. Proceedings IEEE International Conference on Robotics and Automation, 3 (1994), pp. 1916-1921.

[130] Zimmermann, M.: Concurrent Behavior Control - A System's Thinking Approach to Intelligent Behavior. Prof. G. Schweitzer, Prof. M. Hiller, Universität Duisburg. ETH-Dissertation 10022, 1993.

Curriculum Vitae

Surname: Morgado de Gois

First Name: Jorge Audrin

Date of birth: April 24, 1973

Higher Education:

1998 – 1999 M.Sc. Mechanical Sciences,
Military Institute of Engineering – Brazil

1991 – 1995 Mechanical Engineer,
Military Institute of Engineering - Brazil

Career, Employment:

2000 –2001 Undergraduate-Course Docent,
Department of Mechanical Engineering,
Military Institute of Engineering - Brazil

1996 – 1997 Product development Engineer,
Institute for Research and Development
Brazilian Army

Specialization Courses:

Modelling, simulation and control
Electrical Engineering Department,
Military Institute of Engineering - Brazil

Project Management and Production Planning
Institute for Research and Development
Brazilian Army